What Is a Good Domain Description? Evaluating & Revising Action Theories in Dynamic Logic

Ivan Varzinczak

Institut de Recherche en Informatique de Toulouse
CNRS INPT UPS UT1

Ivan José Varzinczak

What Is a Good Domain Description?

Evaluating & Revising Action Theories in Dynamic Logic

A thesis submitted in fulfillment of requirements
for the degree of Doctor in Artificial Intelligence to

Supervisor: Andreas Herzig

October 2006

Typeset in Palatino and Euler by TEX and LATEX 2ε.

Institut de Recherche en Informatique de Toulouse
UNIVERSITÉ PAUL SABATIER – TOULOUSE III
U.F.R. MIG

THÈSE

en vue de l'obtention du grade de
docteur de l'Université Paul Sabatier
Spécialité : Intelligence Artificielle
présentée et soutenue par

Ivan José Varzinczak

le 27 octobre 2006

What Is a Good Domain Description?
Evaluating & Revising Action Theories in Dynamic Logic

Directeur de thèse : Andreas Herzig

Devant le jury :

M. Robert Demolombe	Directeur de Recherche ONERA Toulouse	
M. Olivier Gasquet	Professeur Université Paul Sabatier	(Président)
M. Laurent Perrussel	Maître de Conférence Université Toulouse 1	
Mme. Marie-Christine Rousset	Professeur LSR-IMAG Grenoble	(Rapportrice)
M. Michael Thielscher	Professeur Dresden University of Technology	(Rapporteur)

To Sihem

Acknowledgments

Nobody is so much his own buddy
that he does not need anybody.

— Anonymous

First of all I want to thank Andreas Herzig, not only for accepting me in his group, but also for his extremely valuable supervision and for the unequal human support and life example. Definitely, the process of becoming a researcher goes very far beyond technical discussions.

Special thanks to Marcos Castilho, a great bridge between the far away Contenda and the *ville rose*.

I am grateful to the examiners of the text of this thesis: Guilherme Bittencourt, Marie-Christine Rousset and Michael Thielscher. Their remarks and suggestions helped me a lot in improving the final version of the manuscript. I also would like to thank the other members of the jury for the honor of having them all there.

I want to express my gratitude to the LILaC team and all its members for the very nice environment I could share during these years.

Thanks to Robert Demolombe for interesting discussions on some of the subjects of this thesis, and special thanks to Luis Fariñas del Cerro for his human support and also for the honor of being sent to South Africa to represent our research team in an important meeting.

I want to thank Laurent Perrussel, too, for all our discussions (both funny and technical), and for all his scientific, material and moral support.

I am very grateful to IRIT and all its personal for their sympathy and good work conditions I had the luck to find here.

Thanks to Paul Wong and to the Australian National University for the thesis template and LaTeX styles.

I am grateful to the anonymous referees of JNMR'03 (*Journées Nationales sur les Modèles de Raisonnement*), M4M'03 (Methods for Modalities), ECAI'04 and ECAI'06

(European Conference on Artificial Intelligence), AiML'04 (Advances in Modal Logic), NMR'04 and NMR'06 (Workshop on Non-monotonic Reasoning), IJCAI'05 (International Joint Conference on Artificial Intelligence), and JELIA'06 (European Conference on Logics in Artificial Intelligence) for useful comments on works this thesis rely on. Thanks to Bernardo Cuenca Grau and Rob Miller for useful and interesting discussions about the topics of this work.

I also would like to express all my gratitude to the Brazilian people and to the government of the FEDERATIVE REPUBLIC OF BRAZIL, without whose support (Grant: CAPES BEX 1389/01-7) this work would never have been accomplished.

Lovely thanks to "Aysó", without whose encouragements the first draft of this thesis would not have been written in six days (and I would not be able to rest on the seventh :-)).

Ivan José Varzinczak
Toulouse, October 2006

Ardentes fortuna juvat.

— Virgilius

Contents

List of Figures

Abstract

Traditionally, consistency is the only criterion for the quality of a theory in logic-based approaches to reasoning about actions. This work goes beyond that and contributes to the meta-theory of actions by investigating what other properties a good domain description should satisfy. Having Propositional Dynamic Logic (PDL) as background, we state some meta-theoretical postulates concerning this sore spot. When all postulates are satisfied, we call the action theory *modular*. We point out the problems that arise when the postulates about modularity are violated, and propose algorithmic checks that can help the designer of an action theory to overcome them. Besides being easier to understand and more elaboration tolerant in McCarthy's sense, modular theories have interesting computational properties. Moreover, we also propose a framework for updating domain descriptions and show the importance modularity has in action theory change.

Keywords: Reasoning about actions, modularity, dependence, theory change.

Résumé

Traditionnellement, la consistance est le seul critère pour décider de la qualité d'une théorie dans les approches logiques pour le raisonnement sur les actions. Ce travail va au delà de cela et contribue à la méta-théorie de l'action en proposant d'autres propriétés qu'une bonne description de domaine doit satisfaire. En utilisant la logique dynamique propositionnelle (PDL) comme logique de base, nous énonçons quelques postulats méta-théoriques. Lorsque ces postulats sont satisfaits, nous disons que la théorie d'action est *modulaire*. Nous présentons les problèmes qui surviennent lorsque nos postulats de modularité sont violés, et proposons des algorithmes pour aider le concepteur de la théorie à les résoudre. En plus d'être plus faciles à comprendre et plus tolérantes à l'élaboration au sens de McCarthy, les théories modulaires ont des propriétés intéressantes d'un point de vue computationel. Dans ce travail, nous proposons également une méthode de mise à jour de descriptions de domaine et montrons l'importance de la modularité pour le changement de théories.

Mots-clés : Raisonnement sur les actions, modularité, dépendance, mise à jour de théories.

Resumo

Tradicionalmente, consistência tem sido o único critério de qualidade de teorias em abordagens lógicas para raciocínio sobre ações. O presente trabalho tem por objetivo ir ainda mais longe e contribui com a meta-teoria de ações investigando que outras propriedades uma boa descrição de domínio deve satisfazer. Usando a lógica dinâmica proposicional (PDL) como formalismo de base, enunciamos alguns postulados meta-teóricos. Quando uma dada teoria de ações satisfaz todos os nossos postulados, chamamo-a *modular*. Aqui nós mostramos os problemas que podem surgir quando os postulados de modularidade são violados e igualmente propomos algoritmos que auxiliam o projetista da teoria de ações a solucioná-los. Além de mais fáceis de entender e mais tolerantes à elaboração, no sentido de McCarthy, teorias de ações modulares apresentam também propriedades interessantes do ponto de vista computacional. Além disso, nós aqui também apresentamos operadores para atualização de descrições de domínio, e mostramos a importância da modularidade na modificação de teorias.

Palavras-chave: Raciocínio sobre ações, modularidade, dependência, modificação de teorias.

Chapter 1

where we try to convince the reader this work is worth reading

Introduction

"Well," said Pooh, "what I like best..." and then he had to stop and think. Because although Eating Honey was a very good thing to do, there was a moment just before you began to eat it which was better than when you were, but he didn't know what it was called.

— A.A. Milne, from The House at Pooh Corner

In logic-based approaches to knowledge representation, knowledge concerning a given domain is usually described by logical formulas, also called axioms. A set $\mathcal{T}$ of such formulas is called a (non-logical) *theory*. Theories used in applications are abstractions modeling observed phenomena with the goal of explaining and making predictions about them. That is also the case for reasoning about actions, where we are interested in theories describing the behavior of particular *actions* on properties of the world, called *fluents*. We call such theories *action theories* (alias *domain descriptions*).

Following the tradition in the reasoning about actions community, action theories are collections of statements that have the particular form: "if *context*, then *effect* after *every execution* of *action*"; and "if *precondition*, then *action executable*". The first type of statement is used to express effect laws, i.e., formulas that relates an action with its outcome, given a particular context. The second kind of statement denotes executability laws, those formulas establishing the sufficient conditions under which an action is executable. Their dual gives us the necessary conditions for an action to be executable: "if *precondition*, then *action impossible*". (Such statement can also be seen as a special case of effect laws whose effect is a contradiction.)

Finally, in a representation of a dynamic domain, we also single out statements mentioning no action at all. These can represent laws about the static part of the

world, i.e., the constraints that determine which states are possible, or represent facts observed in a given state. We call the former static laws or domain constraints, while the later are referred to as simple observations.

1.1 What Are Action Theories for?

When describing action theories, the goal is to give a reasoning agent the ability to reason about a dynamic domain and perform rationally in the environment that its action theory models. Hence, action theories are made essentially to perform reasoning with. Among the different types of reasoning an agent can perform when interacting with its environment, we identify:

- Checking consistency of its theory;
- Predicting the effects of actions;
- Explaining the observation of a given effect;
- Establishing a plan to achieve a goal;
- Check the executability or inexecutability of a given action; and
- Revise and update its knowledge about the behavior of an action.

We here briefly discuss about each of such tasks.

Consistency Check

Look at all the sentences which seem true and question them.

— David Reisman

Inconsistent theories are useless outside the realm of paraconsistent logics. Hence, given a theory, an important task is to check its consistency (Figure 1.1).

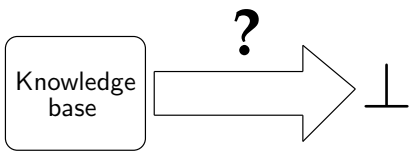

Figure 1.1: Checking consistency of a theory.

That is also the case for theories in reasoning about actions: an agent with inconsistent beliefs about the behavior of actions can perform unpredictably and be unsafe in real world applications. For instance, if the knowledge base of an agent conceived for an on-line flight reservation system becomes inconsistent, the agent may book a flight for a new passenger even if the flight is already full, producing an overbook for the company.

Historically, logical consistency is the most used criterion for evaluating how good a given theory is. Consistency of theories in general has been extensively addressed in the literature on logic-based knowledge representation. In a more or less tacit way, it has also been studied for action theories [9, 99, 74, 96]. More recently, different notions of consistency specific to domain descriptions in reasoning about actions have been proposed [118, 72].

Our main claim in the present work, however, is that mere consistency is not enough to evaluate an action theory. We may have consistent domain descriptions that are not intuitive, and also intuitive theories that, although consistent, may behave unpredictably and be difficult to manage and change. In order to capture these subtleties, something beyond consistency is required. We will come back to this point in the sequel.

Progression, Regression and Plan Generation

When performing reasoning with an action theory, one is naturally interested in doing *progression*, i.e., the prediction of action effects; *regression*, i.e., explaining the state of the world *before* a sequence of actions has taken place; and *planning*, which amounts to finding a sequence of actions whose outcome is the intended goal.

Prediction is very difficult, especially about the future.

— Niels Bohr

Progression (Figure 1.2), also known as *temporal projection*, is the prototypical reasoning problem for dynamic systems. Technically, it is the problem of determining whether a given set of fluents is true after the execution of a sequence of actions. For example, in an on-line booking system, querying the knowledge base whether the flight is booked after the customer has executed the action of paying is an instance of the progression problem.

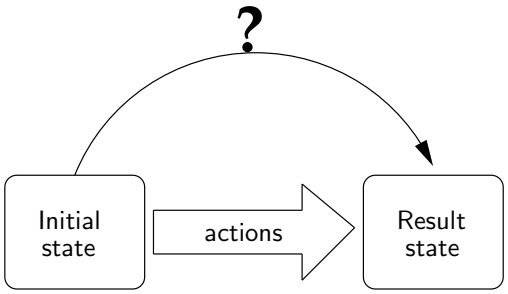

Figure 1.2: Progression: reasoning about the future.

The longer the explanation, the bigger the lie.

— Chinese proverb

Regression (Figure 1.3), also known as *temporal explanation*, consists in finding the set of fluents that hold at the initial situation before a sequence of actions were carried out. In the example above, given that the action of paying has been executed with the result that the flight is now booked, deducing that the client had a valid credit card number is an example of regression.[1]

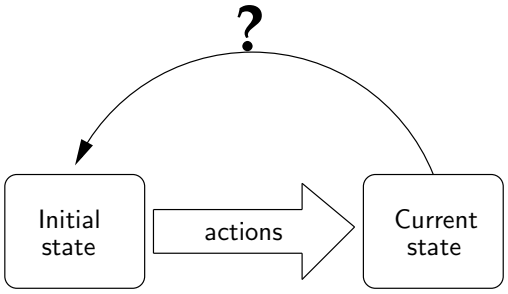

Figure 1.3: Regression: reasoning about the past.

Nothing happens unless first we dream.

— Carl Sandburg

Plan generation (Figure 1.4) is the task of knowing whether there exists a sequence of actions leading to an intended state of the world and, if that is the case, what that sequence is. In our running example, in order to get a flight booked, the agent must

[1] This is an example of *deductive regression* [72].

be able to find the actions necessary to achieve its goal. An associated task is *plan validation*: given a sequence of actions and a goal, decide whether the actions constitute a plan for the goal.

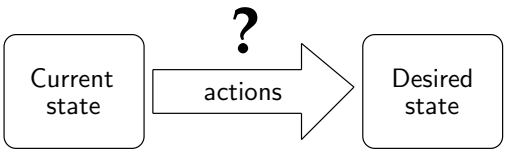

Figure 1.4: Plan generation: what to do to achieve a goal.

Tests for Executability and Inexecutability

Try not. Do, or do not. There is no try.

— Yoda

All tasks of progression, regression and plan generation depend upon whether the involved actions are executable or not. Many approaches in the literature [83, 84, 72] assume that actions are always executable. They follow the so-called "tentativist" approach, according to which one can always attempt to execute an action, whether its outcome is the expected one or not. We here prefer to adopt the "effectivist" approach, in which the execution of an action may fail. This allows us to differentiate *action preconditions*, i.e., the context in which the occurrence of the action is guaranteed, from the action's *effect preconditions*, i.e., the context in which the action, if executable, produces the expected effect. For instance, having a gun is a precondition for shooting, while the gun being loaded is the precondition of the effect that the victim dies.

Then, an important reasoning task in action theories is determining executability/inexecutability of an action in a given context. In our example, the agent must be able to detect that without a credit card number, the action *pay* is not executable (and consequently its effects do not apply).

It turns out that such tasks can have a very high complexity when carried out in formalisms with a minimum of expressivity. One of our goals in this work is to show that we can simplify such a task if we have a theory satisfying some design principles.

Revision and Update of Action Theories

When we are no longer able to change a situation,
we are challenged to change ourselves.

— Viktor Frankl

Just being consistent does not mean that the information coded in an action theory is intuitive. Nor does it mean that even being intuitive it will remain so along the evolution of the world. It is not difficult to conceive action theories describing laws about actions that are completely out of line with respect to the intuitive behavior of the world. In this sense, the agent must be able to revise its beliefs about the behavior of actions. In the same way, it can be the case that the world has just evolved, and then the action theory in the agent's knowledge base is out of date and need thus to be changed. Such situations are depicted in Figure 1.5.

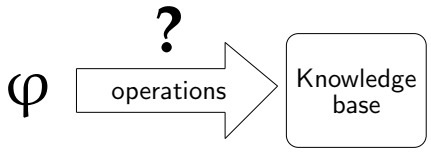

Figure 1.5: Theory change: how to accommodate new information in a knowledge base.

For instance, let the agent in the flight reservation system believe that always after booking a flight to a passenger, this one is confirmed to that flight. Now, if the agent learns that in the case where the flight is full, after booking it the passenger may go to a waiting list, it should be able to make this information fit together with its knowledge about the behavior of booking.

Such cases of theory change are very important when one deals with logical descriptions of dynamic domains: it may always happen that one discovers that an action actually has a behavior that is different from that one has always believed it had. It is important to note, however, that independently of the method to carry out a change in the theory, deciding on its intuition remains a knowledge engineer's task.

1.2 Modular Logic Project

The design of theories in knowledge representation has much more in common with software engineering than one might think. In AI applications, a theory representing a

knowledge base is a piece of software. Hence in the same way as for software projects, one can talk about correctness and evolution of domain descriptions.

Besides that, action theories play an important role when integrated into more complex knowledge representation systems. Those may involve representation and reasoning ability for e.g. knowledge, beliefs, desires and intentions. In order to the components of such an heterogeneous knowledge base fit together, some principles of good design should be considered prior to integrating all those components. Among the principles of the object-oriented paradigm in software development are the following [108, 98]:

1. Work with modules;
2. Minimize interactions between modules;
3. Organize the modules into well-defined layers to help minimize interactions. The goal is to have components of one layer using only components from immediate neighbors, wherever possible; and
4. Anticipate what kind of extensions or modifications might be made in the future, and support this at design time so that one can extend the system with minimal disruption later.

There seems to be an agreement that such principles for object-oriented programming or design are the same as for knowledge representation in general [38, 22, 110, 59] as in reasoning about actions [2, 57, 64, 77]. All the principles above can be applied to the design of domain descriptions, too. We argue that a good domain description should be one whose consistency check and maintenance complexities are minimized, so that any further modification is localized, with a bounded scope. Moreover, we expect that good design of a theory should improve its general performance.

With this in mind, one can see the specification of domain descriptions as a task similar to project development in software engineering: Item 4 above is what has been called *elaboration tolerance* [88]. In this way, a representation is elaboration tolerant to the extent that the effort required to add new information (a new action or effect) to the representation is proportional to the complexity of that information [105]. Items 1, 2 and 3 reflect the concept of *modularity*, which means that different modules should have as few elements as possible in common.

A commonly used guideline in software development is to divide the software into modules, based on their functionality or on the similarity of the information they

handle. This means that instead of having a "jack of all trades" program, it is preferable to split it up into specialized subprograms. For instance, a program made of a module for querying a database and a module for checking its integrity is more modular than a single module that does these two tasks at the same time.

The major benefits of modular systems are reusability, scalability and better management of complexity. Among the criteria commonly used for evaluating how modular a piece of software is are the notions of *cohesion* and *coupling* [98, 108]. Roughly, cohesion is about how well defined a module is, while coupling is about how modules are interdependent. A common sense maxim in object-oriented design is maximize cohesion of modules and diminish their coupling, and this paradigm can also be applied to reasoning about actions [3, 56, 57].

1.3 Objectives and thesis organization

A priori consistency is the only criterion that formal logic provides to check the quality of action theories. Our objective in this work is to go beyond that, and argue that we should require more than the mere existence of a model for a given theory.

Here we claim that all the approaches that are put forward in the literature are too liberal in the sense that we can have satisfiable action theories that are intuitively incorrect. We argue that something beyond the consistency notion is required in order to help us in evaluating a given theory.

Our starting point is the fact that in reasoning about actions one usually distinguishes several kinds of logical formulas. Among these are effect axioms, precondition axioms, and domain constraints. In order to distinguish such non-logical axioms from logical axioms, we prefer to speak of effect laws, executability laws, and static laws, respectively. Moreover we single out those effect laws whose effect is $\bot$ (the contradiction), and call them inexecutability laws.

Given these types of laws, suppose that the language is powerful enough to state conditional effects of actions. For example, suppose that some action a is inexecutable in contexts where φ_1 holds, and executable in contexts where φ_2 holds. It follows that there can be no context where $\varphi_1 \wedge \varphi_2$ holds. Now $\neg(\varphi_1 \wedge \varphi_2)$ is a static law that does not mention a. It is natural to expect that $\neg(\varphi_1 \wedge \varphi_2)$ follows from the set of static laws alone. By means of examples we show that when this is not the case, then unexpected conclusions might follow from the theory $\mathcal{T}$, even in the case that $\mathcal{T}$ is logically consistent.

This motivates postulates requiring that the different laws of an action theory should be arranged modularly, i.e., in separated components, and in such a way that interactions between them are limited and controlled. In essence, we argue that static laws may entail new effects of actions (that cannot be inferred from the effect laws alone), while effect laws and executability laws should never entail new static laws that do not follow from the set of static laws alone. We formulate postulates that make these requirements precise. It will turn out that in all existing accounts that allow for these four kinds of laws [78, 83, 112, 23, 14, 119], consistent action theories can be written that violate these postulates.

We here give algorithms that allow one to check whether an action theory satisfies the postulates we state. With such algorithms, the task of correcting flawed action theories can be made easier.

The ideas we are going to develop in this thesis are not intended as the final word on how action theories should be formalized in reasoning about actions; indeed, they hardly constitute the initial word on how to do that!

The present work is structured as follows: in Chapter 2, we establish the formal background needed to the core of the thesis. Chapter 3 makes a systematic analysis of some modularity approaches when applied to the case of reasoning about actions. In Chapter 4, we propose another view of decomposing a theory into modules, presenting it in a simple framework that abstracts from the frame problem. We then present the solution to the frame problem we will rely on in the rest of this work and shows that it subsumes Reiter's regression technique (Chapter 5). After that, we investigate the behavior of existing solutions to the frame problem, including ours, in more complex scenarios (Chapter 6). In Chapter 7, we revisit our concept of modularity by giving a more fine grained account of it with the solution to the frame problem. We then generalize our modularity principle (Chapter 8) and present the main properties its satisfaction gives us. In Chapter 9, we make a step toward action theory update and present operators for contracting action laws. Before concluding, we make some discussion and address related work in the field (Chapter 10).

Part of the material here presented have appeared earlier elsewhere: Chapter 4 is a joint work with Andreas Herzig that was published as [58]. Chapter 5 is the result of a joint collaboration with Robert Demolombe and Andreas Herzig that appeared in [25, 26]. A preliminary version of Chapter 6 appeared as [55]. Parts of Chapter 7 are an improvement of the preliminary works published in [56]. Chapter 9 is the result of a joint work with Andreas Herzig and Laurent Perrussel appeared in [52] and [51].

Chapter 2

where we get acquainted with the formal

Describing Action Theories

Let these describe the indescribable.

— Lord Byron

In this chapter, we present the logical foundations which will serve as the basis for developing the central ideas of this work. As our base formalism, we have chosen modal logics [60, 16], and we describe action theories in $*$-free PDL, i.e., PDL without the iteration operator $*$. We here establish the ontology of dynamic domains and formally define what an action theory is. For more details on PDL, see [49, 50]; for the benefits of dynamic logic as a formalism for reasoning about actions, see [14, 43, 121].

2.1 Dynamic Logic

Let $\mathfrak{Act} = \{a_1, a_2, \ldots\}$ be the set of all *atomic action constants* of a given domain ($\mathfrak{Act} \neq \emptyset$). Our main running example is in terms of the Walking Turkey Scenario [112], depicted in Figure 2.1. There, the atomic actions are *load*, *shoot* and *tease*. We use a as a variable for atomic actions. To each atomic action a there is an associated modal operator $[a]$. This gives us a multimodal logic [97]. Here we suppose that the underlying multimodal logic is independently axiomatized (i.e., the logic is a fusion and there is no interaction between the modal operators [69, 70]).

Let $\mathfrak{Prop} = \{p_1, p_2, \ldots\}$ denote the set of all *propositional constants*, also called *fluents* or *atoms*. Examples of those are *loaded*, *alive* and *walking*. We use p as a variable for propositional constants.

We suppose from now on that both sets $\mathfrak{Act}$ and $\mathfrak{Prop}$ are finite.

Figure 2.1: The Walking Turkey Scenario.

We use small Greek letters $\varphi, \psi, \ldots$ to denote *classical formulas*, also called *boolean* formulas. They are recursively defined in the following way:

$$\varphi \ ::= p \mid \top \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid \varphi \leftrightarrow \varphi$$

The set of all classical formulas will be denoted by $\mathfrak{Fml}$.

Examples of classical formulas are $\mathit{walking} \rightarrow \mathit{alive}$ and $\neg(\mathit{bachelor} \wedge \mathit{married})$.

Given $\varphi \in \mathfrak{Fml}$, by $\mathit{valuations}(\varphi)$ we denote the set of all propositional valuations making φ true. We view a valuation as a maximally-consistent set of literals. For instance, if $\mathfrak{Prop} = \{\mathit{alive}, \mathit{walking}\}$, then there are four valuations: $\{\mathit{alive}, \mathit{walking}\}$, $\{\mathit{alive}, \neg\mathit{walking}\}$, $\{\neg\mathit{alive}, \mathit{walking}\}$ and $\{\neg\mathit{alive}, \neg\mathit{walking}\}$. A classical formula φ is *classically consistent* if and only if $\mathit{valuations}(\varphi) \neq \emptyset$, i.e., there is at least one valuation in classical propositional logic that makes it true. We denote $\models_{\mathsf{CPL}}$ the standard logical consequence in classical propositional logic.

The set of all literals is $\mathfrak{Lit} = \mathfrak{Prop} \cup \{\neg p \ : \ p \in \mathfrak{Prop}\}$. Examples of literals are *alive* and $\neg\mathit{walking}$. We will use ℓ as a variable for literals. If $\ell = \neg p$, then we identify $\neg\ell$ with p.

A *clause* χ is a disjunction of literals. We say that a literal ℓ *appears* in a clause χ, written $\ell \in \chi$, if ℓ is a disjunct of χ.

We denote complex formulas (possibly with modal operators) by capital Greek letters $\Phi_1, \Phi_2, \ldots$ They are recursively defined in the following way:

$$\Phi \ ::= \varphi \mid [a]\Phi \mid \langle a \rangle \Phi \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Phi \rightarrow \Phi \mid \Phi \leftrightarrow \Phi$$

where Φ denotes a complex formula. The dual operator of $[a]$ is $\langle a \rangle$ and it is defined by: $\langle a \rangle \Phi =_{\text{def}} \neg[a]\neg\Phi$. Sequential composition of actions is defined by the abbreviation $[a_1;a_2]\Phi =_{\text{def}} [a_1][a_2]\Phi$. Examples of complex formulas are $\mathit{loaded} \rightarrow [\mathit{shoot}]\neg\mathit{alive}$ and $\mathit{hasGun} \rightarrow \langle \mathit{load};\mathit{shoot} \rangle (\neg\mathit{alive} \wedge \neg\mathit{loaded})$.

If $\mathcal{T}$ is a set of formulas (modal or classical), $atm(\mathcal{T})$ returns the set of all atoms occurring in $\mathcal{T}$. For instance, $atm(\{\neg\neg\neg p_1, [a]p_2\}) = \{p_1, p_2\}$.

For parsimony's sake, whenever there is no confusion we identify a set of formulas with the conjunction of its elements. The semantics is that for multimodal K [97, 10].

Definition 2.1 (PDL**-model)**
A PDL*-model is a tuple $\mathscr{M} = \langle W, R \rangle$ where W is a set of valuations (alias possible worlds), and $R : \mathfrak{Act} \longrightarrow 2^{W \times W}$ a function mapping action constants a to accessibility relations $R_a \subseteq W \times W$.*

As an example, for $\mathfrak{Act} = \{a_1, a_2\}$ and $\mathfrak{Prop} = \{p_1, p_2\}$, we have the PDL-model $\mathscr{M} = \langle W, R \rangle$, where

$$W = \{\{p_1, p_2\}, \{p_1, \neg p_2\}, \{\neg p_1, p_2\}\},$$

$$R(a_1) = \left\{ \begin{array}{c} (\{p_1, p_2\}, \{p_1, \neg p_2\}), (\{p_1, p_2\}, \{\neg p_1, p_2\}), \\ (\{\neg p_1, p_2\}, \{\neg p_1, p_2\}), (\{\neg p_1, p_2\}, \{p_1, \neg p_2\}) \end{array} \right\}$$

$$R(a_2) = \{(\{p_1, p_2\}, \{p_1, \neg p_2\}), (\{p_1, \neg p_2\}, \{p_1, \neg p_2\})\}$$

Figure 2.2 gives a graphical representation of $\mathscr{M}$.

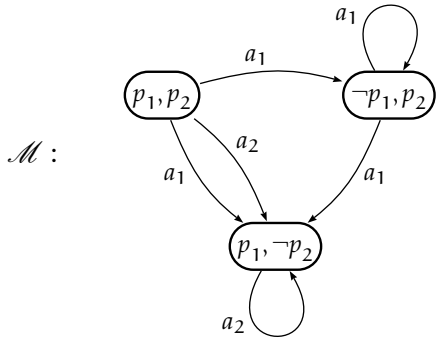

Figure 2.2: Example of a PDL-model for $\mathfrak{Act} = \{a_1, a_2\}$, and $\mathfrak{Prop} = \{p_1, p_2\}$.

Given $\mathscr{M} = \langle W, R \rangle$, $a \in \mathfrak{Act}$, and $w, w' \in W$, we write R_a instead of $R(a)$, and wR_aw' instead of $w' \in R_a(w)$.

Definition 2.2 (PDL truth conditions)
Given a PDL*-model* $\mathscr{M} = \langle W, R \rangle$*, the satisfaction relation is defined as the smallest relation satisfying:*

- $\models^{\mathscr{M}}_{w} p$ *(*p *is true at world* w *of model* $\mathscr{M}$*) if* $p \in w$*;*
- $\models^{\mathscr{M}}_{w} [a]\Phi$ *if for every* w' *such that* wR_aw'*,* $\models^{\mathscr{M}}_{w'} \Phi$*; and*
- *the usual truth conditions for the other connectives.*

Definition 2.3 (Model of formulas)
A PDL*-model* $\mathscr{M}$ *is a model of* Φ *(noted* $\models^{\mathscr{M}} \Phi$*) if and only if for all* $w \in W$*,* $\models^{\mathscr{M}}_{w} \Phi$*.* $\mathscr{M}$ *is a model of a set of formulas* $\mathcal{T}$ *(noted* $\models^{\mathscr{M}} \mathcal{T}$*) if and only if* $\models^{\mathscr{M}} \Phi$ *for every* $\Phi \in \mathcal{T}$*.*

In the model depicted in Figure 2.2, we have $\models^{\mathscr{M}} p_1 \rightarrow [a_2]\neg p_2$ and $\models^{\mathscr{M}} p_1 \vee p_2$.

Definition 2.4 (Global consequence)
A formula Φ *is a* consequence of the set of global axioms $\mathcal{T}$ *in the class of all* PDL*-models (noted* $\mathcal{T} \models_{\mathsf{PDL}} \Phi$*) if and only if for every* PDL*-model* $\mathscr{M}$*, if* $\models^{\mathscr{M}} \mathcal{T}$*, then* $\models^{\mathscr{M}} \Phi$*.*[1]

We suppose that the logic under consideration is *compact* [33].

Having established the formal substratum our presentation will rely on, we present in the next section the different types of formulas we will henceforth use to describe dynamic domains.

2.2 Describing the Behavior of Actions in PDL

Before elaborating a theory, we need to specify what we are about to describe, i.e., what the formulas are supposed to interpret. Following the tradition in the literature, we identify a domain (alias scenario) with the actions we take into account and the fluents they can change. More formally, we have:

Definition 2.5 (Domain signature)
A domain signature *is a tuple* $\langle \mathfrak{Act}, \mathfrak{Prop} \rangle$*.*

An example of a domain signature (domain, for short) is the well-known Yale Shooting Scenario [47], whose signature comprises the actions *load*, *wait* and *shoot*, and fluents *loaded* and *alive*.

[1] Instead of global consequence, in [14] local consequence is considered. For that reason, a further modal operator $\Box$ had to be introduced, resulting in a logic which is multimodal K plus monomodal S4 for $\Box$, and where axiom schema $\Box\Phi \rightarrow [a]\Phi$ holds.

The beginning of wisdom is to call things by their right names.

— Chinese proverb

Given a domain $\langle \mathfrak{Act}, \mathfrak{Prop} \rangle$, we are interested in theories whose statements describe the behavior of actions of $\mathfrak{Act}$ on the fluents of $\mathfrak{Prop}$. PDL allows for the representation of such statements, that we here call *action laws*. We distinguish several types of them. We call *effect laws* formulas relating an action to its effects. Statements of conditions under which an action cannot be executed are called *inexecutability laws*. *Executability laws* in turn stipulate the context where an action is guaranteed to be executable. Finally, *static laws* are formulas that do not mention actions. They express constraints that must hold in every possible state. These four types of laws are our fundamental entities and we introduce them more formally in the sequel.

Static Laws

Frameworks which allow for indirect effects of actions make use of logical formulas that state invariant propositions about the world. Such formulas delimit the set of possible states. They do not refer to actions, and we suppose here that they are expressed as formulas of classical propositional logic.

Definition 2.6 (Static law)
A static law[2] *is a formula* $\varphi \in \mathfrak{Fml}$.

In our running example, the static law *walking* $\rightarrow$ *alive* says that if a turkey is walking, then it must be alive. Another one is $saved \leftrightarrow (mbox1 \vee mbox2)$, which states that an e-mail message is saved if and only if it is in mailbox 1 or in mailbox 2 or in both [15].

In some action languages, such as $\mathcal{AR}$ [65, 44] for example, we would write the statement always *alive* $\rightarrow$ *walking*, and in a Situation Calculus [90] variant, it would be the first-order formula

$$\forall s.(Holds(walking, s) \rightarrow Holds(alive, s)).$$

[2] In the literature, static laws are often called *domain constraints* or *integrity constraints*. Because the different laws for actions that we shall introduce in the sequel could in principle also be called like that, we avoid these terms.

At first glance, no requirement concerning consistency of the static laws is made. Of course, we want them to be consistent, otherwise the whole theory is inconsistent. As we are going to see in the sequel, however, consistency of the static laws alone is not enough to guarantee the consistency and even the intuitiveness of an action theory as a whole.

Effect Laws

Logical frameworks for reasoning about actions contain expressions linking actions and their effects. We suppose that such effects might be conditional, and thus get a third component of such laws.

In PDL, the formula $[a]\Phi$ states that formula Φ is true after every possible execution of action a.

Definition 2.7 (Effect law)
An effect law[3] for action a *is of the form* $\varphi \rightarrow [a]\psi$, *where* $\varphi, \psi \in \mathfrak{Fml}$, *with* ψ *classically consistent.*

The consequent ψ is the effect which obtains when action a is executed in a state where the antecedent φ holds. An example of an effect law is $loaded \rightarrow [shoot]\neg alive$, saying that whenever the gun is loaded, after shooting, the turkey is dead. Another one is $\top \rightarrow [tease]walking$: in every circumstance, the result of teasing is that the turkey starts walking. For parsimony's sake, the latter effect law will be written $[tease]walking$.

Note that the consistency requirement for ψ makes sense: if ψ is inconsistent, then we have an inexecutability law, that we consider as a separate entity and which we are about to introduce formally in the sequel. On the other hand, if φ is inconsistent, then the effect law is obviously superfluous.

For the first example above, in action languages one would write the statement

$$shoot \text{ causes } \neg alive \text{ if } loaded,$$

and in the Situation Calculus formalism one would write the first-order formula

$$\forall s.(Holds(loaded, s) \rightarrow \neg Holds(alive, do(shoot, s))).$$

[3] Effect laws are often called *action laws*, but we prefer not to use that term here because it would also apply to executability laws that are to be introduced in the sequel.

Inexecutability Laws

We consider effect laws with inconsistent consequents as a particular kind of law which we call inexecutability laws. (Such laws are sometimes called qualifications [85].) This allows us to avoid mixing things that are conceptually different: for an action a, an effect law mainly associates it with a consequent ψ, while an inexecutability law only associates it with an antecedent φ, viz. the context which precludes the execution of a.

Definition 2.8 (Inexecutability law)
An inexecutability law for action a *is of the form* $\varphi \rightarrow [a]\bot$, *where* $\varphi \in \mathfrak{Fml}$.

For example, $\neg \mathit{hasGun} \rightarrow [\mathit{shoot}]\bot$ expresses that action *shoot* cannot be executed if the agent has no gun. Another example is $\mathit{dead} \rightarrow [\mathit{tease}]\bot$: a dead turkey cannot be teased.

In $\mathcal{AR}$ we would write the statement

$$\textsf{impossible } \mathit{shoot} \textsf{ if } \neg \mathit{hasGun},$$

and in the Situation Calculus, our example would be

$$\forall s.(\neg \mathit{Holds}(\mathit{hasGun}, s) \rightarrow \neg \mathit{Poss}(\mathit{shoot}, s)).$$

Executability Laws

With only static and effect laws one cannot guarantee that the action *shoot* can be executed whenever the agent has a gun. We need thus a way to state such conditions.

In dynamic logic, the dual $\langle a \rangle \varphi$, defined as $\neg[a]\neg\varphi$, can be used to express executability. The formula $\langle a \rangle \top$ thus reads "execution of action a is possible".

Definition 2.9 (Executability law)
An executability law for action a *is of the form* $\varphi \rightarrow \langle a \rangle \top$, *where* $\varphi \in \mathfrak{Fml}$.

For instance, $\mathit{hasGun} \rightarrow \langle \mathit{shoot} \rangle \top$ says that shooting can be executed whenever the agent has a gun, and $\top \rightarrow \langle \mathit{tease} \rangle \top$, also written $\langle \mathit{tease} \rangle \top$, establishes that the turkey can always be teased.

Some approaches (most prominently Reiter's [99, 100]) use biconditionals of the form $\varphi \leftrightarrow \langle a \rangle \top$, called *precondition axioms*. This is equivalent to $\neg\varphi \leftrightarrow [a]\bot$, which highlights that they merge information about inexecutability with information about executability. Here we consider these entities to be different and keep them separate.

In action languages in general, such laws are not represented, they are rather implicitly inferred from inexecutability statements (cf. Section 8.2). In the Situation Calculus, our example would be stated as

$$\forall s.(Holds(hasGun, s) \rightarrow Poss(shoot, s)).$$

Whereas all the extant approaches in the literature that allow for indirect effects of actions contain static and effect laws, and provide a way for representing inexecutabilities (in the form of implicit qualifications [42, 78, 112]), the status of executability laws is less consensual. Some authors [102, 28, 83, 112] more or less tacitly consider that executability laws should not be made explicit but rather inferred by the reasoning mechanism. Others [78, 23, 14, 119] have executability laws as first class objects one can reason about.

It seems a matter of debate whether one can always do without executabilities. In principle, it seems to be strange to just state information about necessary conditions for action execution (inexecutabilities) without saying anything about its sufficient conditions. This is the reason why we think that we need executability laws. Indeed, in several domains one wants to explicitly state under which conditions a given action is guaranteed to be executable, e.g. that a robot never gets stuck and is always able to execute a move action. And if we have a plan such as $load; shoot$ (*load* followed by *shoot*) of which we know that it achieves the goal $\neg alive$, then we would like to be sure that it is executable in the first place![4] In any case, allowing for executability laws gives us more flexibility and expressive power.

2.3 Action Theories

An ounce of action is worth a ton of theory.

— Ralph Waldo Emerson

Given a domain $\langle \mathfrak{Act}, \mathfrak{Prop} \rangle$, let $\mathcal{L}$ denote the *language* of our formalism, i.e., all well formed sentences of the logic under consideration built upon the objects in the signature and the logical connectives. Let $\mathcal{T}$ be the *theory* (set of non-logical axioms) describing the behavior of the actions of the domain, i.e., $\mathcal{T}$ is a set of global axioms in

[4] Of course, this would require a solution to the qualification problem [85].

Fitting's sense [33] of the types defined above. Let $\mid\!\approx$ be a *consequence relation* (possibly nonmonotonic) defined on $\mathcal{L}$. We thus define action theories:

Definition 2.10 (Action theory)
An action theory *(alias domain description) is a tuple $\mathcal{D} = \langle \mathcal{L}, \mid\!\approx, \mathcal{T} \rangle$, where $\mathcal{L}$ is a language, $\mathcal{T}$ a set of formulas of $\mathcal{L}$, and $\mid\!\approx$ a consequence relation defined on $\mathcal{L}$.*

As an example of an action theory, consider $\mathcal{D}_{wts} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$, where $\mathcal{L}_{\mathsf{PDL}}$ is the set of all PDL-formulas, $\models_{\mathsf{PDL}}$ is the consequence relation in PDL (cf. Definition 2.4), and the theory $\mathcal{T}$ is given by:

$$\mathcal{T} = \left\{ \begin{array}{c} \mathit{walking} \rightarrow \mathit{alive}, \neg \mathit{loaded} \rightarrow [\mathit{load}]\mathit{loaded}, \\ \mathit{loaded} \rightarrow [\mathit{shoot}]\neg \mathit{alive}, \mathit{hasGun} \rightarrow \langle \mathit{shoot} \rangle \top, \\ \neg \mathit{hasGun} \rightarrow [\mathit{shoot}]\bot, [\mathit{tease}]\mathit{walking}, \\ \langle \mathit{tease} \rangle \top, \langle \mathit{load} \rangle \top \end{array} \right\}$$

Then $\mathcal{D}_{wts}$ is an action theory in PDL formalizing the Walking Turkey Scenario [112]. Figure 2.3 below shows a PDL-model for the theory component of the domain description above.

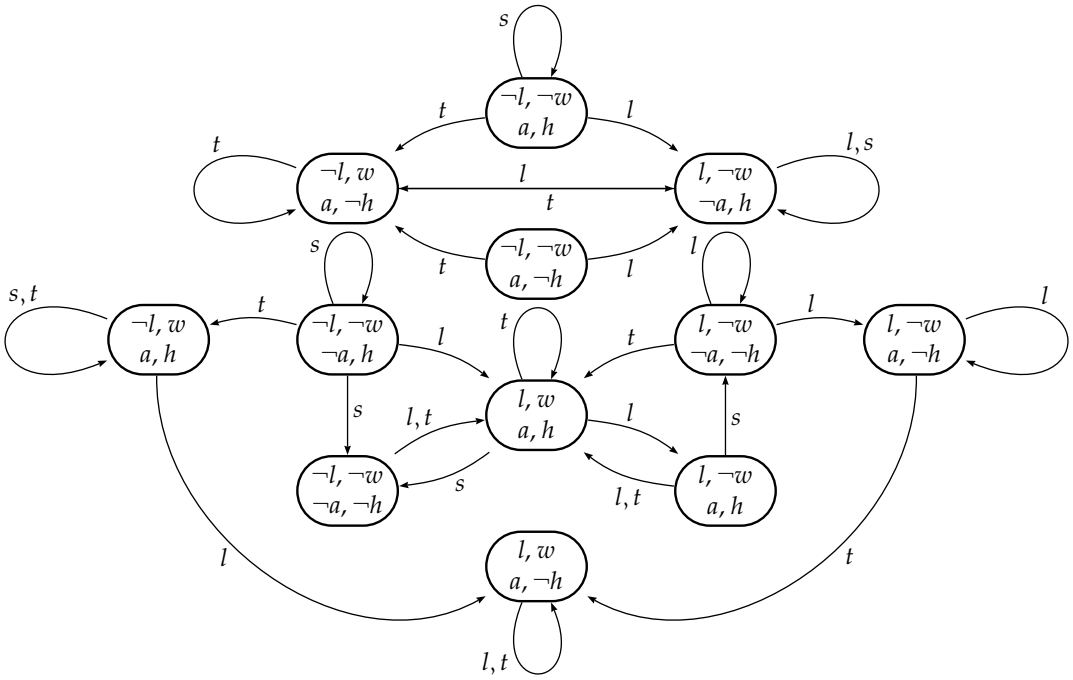

Figure 2.3: A model for the Walking Turkey Scenario: *l*, *w*, *a*, and *h* stand for, respectively, *loaded*, *walking*, *alive* and *hasGun*. Actions *shoot*, *tease* and *load* were abbreviated, respectively, to *s*, *t* and *l*.

Definition 2.11 (Action theory entailment)
Let $\mathcal{D} = \langle \mathcal{L}, \mathrel{|\!\approx}, \mathcal{T} \rangle$ be an action theory, and $\Phi \in \mathcal{L}$. $\mathcal{D}$ entails *Φ (noted $\mathcal{D} \models \Phi$) if and only if $\mathcal{T} \mathrel{|\!\approx} \Phi$.*

(To avoid confusion, we remember that we denote entailment in classical propositional logic by $\models_{\mathsf{CPL}}$.)

In our action theory example, we have $\mathcal{D}_{wts} \models \mathit{loaded} \rightarrow [\mathit{shoot}]\neg \mathit{walking}$ and $\mathcal{D}_{wts} \models [\mathit{tease}]\mathit{alive}$.

Let $Cn(\mathcal{D}) = \{\Phi \ : \ \mathcal{D} \models \Phi\}$ denote the set of all consequences of action theory $\mathcal{D}$. We define when two action theories are equivalent:

Definition 2.12 (Action theory equivalence)
Action theories $\mathcal{D}_1$ and $\mathcal{D}_2$ are equivalent *if and only if $Cn(\mathcal{D}_1) = Cn(\mathcal{D}_2)$.*

In the rest of this work, we analyze the design of action theories as defined here and see how difficult it can be to achieve the desired intuition.

Chapter 3

where we take a look at the grass in the neighbors' garden

Modularity in Reasoning about Actions

To know the road ahead, ask those coming back.

— Chinese proverb

We here identify two main trends on modularity of descriptions: one pragmatic, programming language driven, and one logical theoretic driven. We point out that both proposals are inadequate as accounts of modularity when applied to theories in reasoning about actions. We show that they are either too weak or too strong and do not completely avoid unwanted interactions between modules. We also claim that modules designed following their directives may be as complex as whole theories.

3.1 The Need for Modules

Modularity has become one of the words of order in many areas of software development. That is also the case for knowledge representation and reasoning, where monolithic descriptions have shown to be of high complexity for dealing with.

The last years have seen the flourish of plenties of papers [2, 46, 68, 57, 64] that in a more or less tacit way talk about concepts as modules, reusability, intelligibility, evaluation, maintainability, independence and self-content, elaboration tolerance and many others. Most of these terms are borrowed from software engineering, sometimes without a clear notion of the impacts that they can have when transplanted to domains where their use is not a matter of intuition, but rather they have to accommodate with

well established formal settings. Here we point out that this is not a simple task, especially when logic is the formal substratum in which knowledge is represented.

Despite the apparent fragility of the well-known toy scenarios commonly used in this domain to illustrate typical problems in the area, things get more serious when we move to the "real" world. One can expect that action theories describing the behavior of actions for applications of real interest will be of very high complexity. By this we mean amount of information being represented, the internal relationship among data, the feasibility of inferences in a huge set of formulas, as well as the difficulty for future amendments.

Thus, the question that naturally arises is "how can we ease the knowledge engineer's task in describing a domain"? One answer, of course, following the divide-and-conquer trend, is "modularizing the action theory". But what does it really mean to modularize an action theory? For that we give a (general) definition of a module prototype.

Definition 3.1 (Module prototype)
A module prototype *of an action theory* $\mathcal{D} = \langle \mathcal{L}, \approx, \mathcal{T} \rangle$ *is a description* $\mathcal{D}' = \langle \mathcal{L}', \approx', \mathcal{T}' \rangle$ *such that* $\mathcal{L}' \subseteq \mathcal{L}$, $\approx' \subseteq \approx$ *and* $\mathcal{T}' \subseteq \mathcal{T}$.[1]

A module prototype is just a syntactic-based fragment of a description $\mathcal{D}$. Like in structural and object-oriented programming, to modularize an action theory is not just a matter of cutting the description in a whole bunch of slices. Such a decoupling must be done so that the resulting theory has interesting properties regarding the above requirements. We are going to see in the rest of this work that to be really considered as a module, pieces of descriptions are usually required to satisfy some desiderata.

We can find in the literature several proposals on modularization of action theories that are quite close to software engineering and object-oriented programming. Some examples are the object-oriented first-order logic (OOFOL [1]) and its Situation Calculus variant [2], Gustafsson and Kvarnström's framework for elaboration tolerance [46], and Lifschitz and Ren's modular action description language [77]. Despite the well developed formal background, such approaches are more focused on the implementation level (which is of course important) and either do not take into account or make too restrictive assumptions about in order to get rid of an important issue when describing a domain: unforeseen interactions between modules, or even between components of a single module.

[1] Module prototypes are thus seen as sub-descriptions, and action theories in our sense are themselves module prototypes.

In the same way, we see good work in the logician's community concerning modularity (or similar notions) of logical theories in general [38], and of theories in description logics [22]. However, as we are going to see, when bringing such definitions to the case of reasoning about actions, we get a too restrictive notion of modularity with which either there is no way to modularize a description or the modules are difficult to understand.

3.2 OO-driven Logical Modularity

Regarding the titles of this and the next section, we do not want to say that the approaches we analyze here are not logical. We just have put them apart because they are more engineering-oriented, in the sense that their respective formalisms have been mainly developed with the aim of serving as engineer tools.

There are several proposals on modularization of action theories that are quite close to software engineering and object-oriented programming [2, 46, 77]. The main feature of these approaches is the decomposition of descriptions in a way similar to that programmers usually do in decomposing software applications. Given a domain, their parts are associated with sub-domains. Action theories are thus composed of sub-descriptions that interact in some way, e.g. by sharing common information, inheriting properties [46, 77], or message passing [5, 6].

We here take the OOFOL formalism [1] and its Situation Calculus flavor [2] as our guiding paradigm in this section. The reason is that it is representative of this category, and the fact of being oriented to reasoning about actions will ease further comparisons.

Amir [2, 4] focuses on design and maintainability of domain descriptions applying many of the concepts of the object-oriented paradigm in the Situation Calculus. In that work, guidelines for a partitioned representation of a given theory are presented, with which the inference task can also be optimized [5, 4, 6], as it is restricted to the part of the theory that is relevant to a given query. This is observed specially when different agents are involved: the design of an agent's theory can be done with no regard to others', and after the integration of multiple agents, queries about an agent's beliefs do not take into account the belief state of other agents. Such a feature of a description is called *conditional independence* [2].

The original approach is first-order, but we here present it using the syntax of PDL, which has no harm on its basic intuitions.

In the OOFOL approach, an action theory $\langle \mathcal{L}, \approx, \mathcal{T} \rangle$ is decomposed in module prototypes $\langle \mathcal{L}_1, \approx, \mathcal{T}_1 \rangle, \ldots, \langle \mathcal{L}_n, \approx, \mathcal{T}_n \rangle$ such that

- $\mathcal{L}_i \subseteq \mathcal{L}$ is a PDL language ;
- for every $1 \leq i \leq n$, $\mathcal{T}_i = \langle A_i, I_i \rangle$, where A_i is a set of formulas (axioms) such that $\mathcal{L}(A_i) = \mathcal{L}_i$ ($\mathcal{L}_i$ contains only the symbols appearing in formulas of A_i), and $I_i \subseteq \{\ell \ : \ \ell \in \mathcal{L}_i\}$ is the module's *interface*, i.e., the literals it shares with other modules; and
- $\mathcal{T} = \bigcup_{1 \leq i \leq n} A_i$.

Sub-descriptions are thus seen as objects in the object-oriented sense[2], each one having its own data (the set of axioms A_i) and a communication link with other objects (its interface I_i). Two distinct objects do not necessarily need to have distinct languages, unless they are completely disconnected, i.e., atoms or actions in one of them are never taken into account in the others' inference. Given two distinct objects that are supposed to interact, their interface links establish (semantical) equality or equivalence between symbols in their respective languages and are restricted to only the symbols appearing in the interface (see below).

For the case of reasoning about actions, each $\mathcal{T}_i$ is designed so that the respective A_i component contains formulas of a specific type, i.e., descriptions are partitioned into a module for effect laws, a module for static laws, etc. As an example, the action theory $\mathcal{D}_{wts} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$ formalizing the Walking Turkey Scenario (cf. Section 2.3) would be decomposed in

$$\mathcal{D}_{wts1} = \langle \mathcal{L}_1, \models_{\overline{\mathsf{CPL}}}, \langle \{walking_1 \rightarrow alive_1\}, \{walking_1, alive_1\} \rangle \rangle$$

$$\mathcal{D}_{wts2} = \langle \mathcal{L}_2, \models_{\overline{\mathsf{PDL}}}, \langle \left\{ \begin{array}{c} \top \leftrightarrow \langle tease_2 \rangle \top, \\ hasGun_2 \leftrightarrow \langle shoot_2 \rangle \top \end{array} \right\}, \emptyset \rangle \rangle$$

$$\mathcal{D}_{wts3} = \langle \mathcal{L}_3, \models_{\overline{\mathsf{PDL}}}, \langle \left\{ \begin{array}{c} \neg loaded_3 \rightarrow [load_3] loaded_3, \\ loaded_3 \rightarrow [shoot_3] \neg alive_3, \\ [tease_3] walking_3 \end{array} \right\}, \{walking_3, alive_3\} \rangle \rangle$$

together with the equivalences $\models_{\overline{\mathsf{CPL}}} walking_1 \leftrightarrow walking_3$, and $\models_{\overline{\mathsf{CPL}}} alive_1 \leftrightarrow alive_3$, and the equalities $tease_2 = tease_3$ and $shoot_2 = shoot_3$. These say, e.g. that $walking_1$

[2] Do not confound with objects in the domain signature. In object-oriented programming, an object, roughly speaking, is an instance of a class that models an entity of the world [108, 98].

in object $\mathcal{D}_{wts1}$ should be understood as having the same semantics as $walking_3$ in object $\mathcal{D}_{wts3}$, and action $shoot_2$ in object $\mathcal{D}_{wts2}$ should be interpreted as $shoot_3$ in object $\mathcal{D}_{wts3}$. This means, for example, that inferences regarding $walking_1$ in $\mathcal{D}_{wts1}$ also concern $\mathcal{D}_{wts3}$. (For more details on how reasoning is carried out in descriptions that are decomposed that way, see [5, 6]. We here concentrate only in the modeling aspect and the impact it has on what we expect from modules.)

Notice the modifications that we had to carry out with respect to the original formulas in $\mathcal{D}_{wts}$ in order to decompose it with the method defined in [2]. Executability laws and inexecutability laws are mixed together: $hasGun \rightarrow \langle shoot \rangle \top$ and $\neg hasGun \rightarrow [shoot]\bot$ have been combined in the biconditional $hasGun \leftrightarrow \langle shoot \rangle \top$. This is reminiscent of the principle of maximization of executabilities commonly used in the literature [78, 44]. We argue (cf. Section 8.2) that such assumption gives us less flexibility in the design of dynamical systems.

If we want a better decomposed description, we should rather have defined

$$\mathcal{D}_{wts1'} = \langle \mathcal{L}_1, \models_{\overline{\mathsf{CPL}}}, \langle \{walking_1 \rightarrow alive_1\}, \{walking_1, alive_1\} \rangle \rangle$$

$$\mathcal{D}_{wts2'} = \langle \mathcal{L}_2, \models_{\overline{\mathsf{PDL}}}, \langle \left\{ \begin{array}{c} \langle tease_2 \rangle \top, \\ hasGun_2 \rightarrow \langle shoot_2 \rangle \top \end{array} \right\}, \emptyset \rangle \rangle$$

$$\mathcal{D}_{wts3'} = \langle \mathcal{L}_3, \models_{\overline{\mathsf{PDL}}}, \langle \left\{ \begin{array}{c} \neg loaded_3 \rightarrow [load_3] loaded_3, \\ loaded_3 \rightarrow [shoot_3] \neg alive_3, \\ [tease_3] walking_3 \end{array} \right\}, \{walking_3, alive_3\} \rangle \rangle$$

$$\mathcal{D}_{wts4'} = \langle \mathcal{L}_4, \models_{\overline{\mathsf{PDL}}}, \langle \{\neg hasGun_4 \rightarrow [shoot_4]\bot\}, \emptyset \rangle \rangle$$

with $\models_{\overline{\mathsf{CPL}}} walking_1 \leftrightarrow walking_3$ and $\models_{\overline{\mathsf{CPL}}} alive_1 \leftrightarrow alive_3$, and the equalities $tease_2 = tease_3$ and $shoot_2 = shoot_3 = shoot_4$.

In order to correctly make inferences in such a description, it has to take into account a solution to the frame problem [90] (cf. Chapter 4). In [2] this is done by providing another object containing Successor State Axioms [99] connected with the modules above. We do not show this explicitly here and just assume the above description together with such a solution provides a way for deriving all frame axioms. Then, because the *tease* action does not change the status of literal $\neg alive$, from the above theory with its respective solution to the frame problem, we are able to derive the frame axiom $\neg alive \rightarrow [tease]\neg alive$. Because we have $[tease]walking$ and $walking \rightarrow alive$, we also conclude $[tease]alive$. Joining these results gives us the implicit inexecutability [42]

$\neg alive \rightarrow [tease]\bot$. That is an intuitive result. However, with this and the executability $\langle tease\rangle\top$, we conclude *alive*: the turkey never dies (Figure 3.1)!

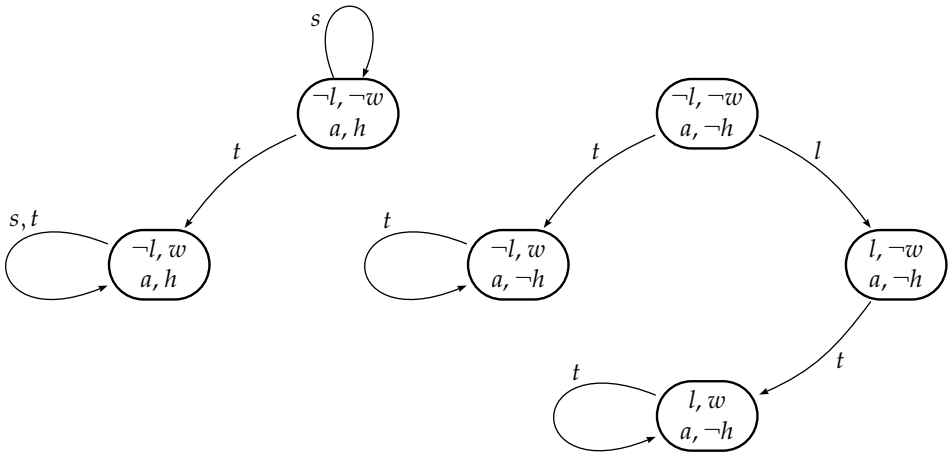

Figure 3.1: A model of the immortal turkey.

The way the proposal in [2] gets rid of such a problem is by imposing a syntactical condition on the antecedents of executabilities and effect laws in order to preclude them of getting in conflict. Roughly speaking, whenever there is an inexecutability $\varphi \rightarrow [a]\bot$ and an executability $\varphi' \rightarrow \langle a\rangle\top$, then $\varphi \wedge \varphi'$ is inconsistent. So, in order to have a safe description, we should change $\mathcal{D}_{wts2}$ in the following way:

$$\mathcal{D}_{wts2''} = \langle \mathcal{L}_2, \models_{\mathsf{PDL}}, \langle \left\{ \begin{array}{c} alive_2 \leftrightarrow \langle tease_2\rangle\top, \\ hasGun_2 \leftrightarrow \langle shoot_2\rangle\top \end{array} \right\}, \{alive_2\}\rangle\rangle$$

That is to say, decomposing the description in its more elementary entities like we did above is not allowed.

3.3 Strong Logic-driven Modularity

Some researchers have tried to capture what modularity in formal logic means [38, 115, 114, 22] at an elementary level. Here we focus on the works of Garson [38] and Cuenca Grau and colleagues [22].

Inspired by Fodor's claims [34], Garson seems to have been the precursor of proposing a notion of modularity in logical systems. In his work, he has given an account of modularity motivated especially by issues as correctness and efficiency of a reasoning system.

In Garson's approach, in order to be a module, a module prototype (alias sub-description) must satisfy two properties:

1. Local correctness: every formula entailed by the sub-description is also entailed by the whole description.

2. Local completeness: every formula in the scope of the sub-description that is entailed by the whole description is also entailed in the sub-description alone. (A formula Φ is in the scope of the module $\mathcal{D}_i = \langle \mathcal{L}_i, \approx, \mathcal{T}_i \rangle$ if $\Phi \in \mathcal{L}_i$.)

Local correctness requires the module prototypes to be "smaller" than the original description, i.e., given $\mathcal{D} = \langle \mathcal{L}, \approx, \mathcal{T} \rangle$ and $\mathcal{D}' = \langle \mathcal{L}', \approx', \mathcal{T}' \rangle$ a module prototype of $\mathcal{D}$, we must have $\mathcal{L}' \subseteq \mathcal{L}$, $\approx' \subseteq \approx$ and $\mathcal{T}' \subseteq \mathcal{T}$. (This is indeed our definition of module prototype, cf. Definition 3.1. In practice, we should claim for the strict inclusion $\subset$, since in modularizing a description we generally do not expect to get the original description as a result.) To see the need for such a property and the motivation behind our definition, if at least one of these inclusions does not hold, then the sub-description can prove more things than the whole description, contradicting the intuition of the concept of module.

Let $\mathcal{D} = \langle \mathcal{L}, \approx, \mathcal{T} \rangle$ be an action theory and $\mathcal{D}_1 = \langle \mathcal{L}_1, \approx, \mathcal{T}_1 \rangle, \ldots, \mathcal{D}_n = \langle \mathcal{L}_n, \approx, \mathcal{T}_n \rangle$ be module prototypes of $\mathcal{D}$. Local completeness states that

$$\mathcal{T} = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_n, \text{ such that } Cn(\mathcal{D}) = \bigcup_{1 \leq i \leq n} Cn(\mathcal{D}_i)$$

and

$$Cn(\mathcal{D}_i) \cap Cn(\mathcal{D}_j) = \emptyset, \forall i, j, i \neq j$$

In other words, the collection of all logical modules should be a kind of 'partition' of the original logical theory.

It is not difficult to see that such a notion of modularity in its own is too strong. First, because each module's theory $\mathcal{T}_i$ by definition entails all logical tautologies. Second, because it holds only for consistent descriptions: it may be the case that an inconsistent domain description has no module that is itself inconsistent, and then there can be formulas entailed by the whole description that are not entailed in their respective module, violating local completeness.

Both these problems have been addressed in [38] and [22]. We can relax local completeness by considering only substantive entailments of the theory, i.e., non-

tautological ones. In what concerns consistency, Garson argues that classical logic is not a good setting for an account of modularity. Besides the complexity of consistency check, the very main reason for that, he says, is the principle of explosion:[3] in classical logic, a contradiction entails any sentence, which makes consistency check very costly. As a manner of overcoming that and guaranteeing local completeness even for inconsistent descriptions, Garson proposes to use relevant logic [30] instead of classical logic. Cuenca Grau and colleagues, on the other hand, rely on the tractable consistency check methods for description logics [8] and do not care about the principle of explosion.

Nevertheless, even relaxing local completeness, if we apply such a notion of modularity to domain descriptions in reasoning about actions, we can have some annoyances. To witness, consider the following example (we illustrate with PDL, but it could also be adapted to other frameworks in the literature that allow for the four types of laws that we use to describe dynamic domains): suppose a domain with, say, two actions a_1 and a_2, and only one atom p. Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$, with $\mathcal{T} = \{p \rightarrow [a_1]\bot, p \rightarrow \langle a_1 \rangle \top, \langle a_2 \rangle \top\}$. Notice that $\mathcal{D}$ is consistent. So, because actions a_1 and a_2 are independent, i.e., they do not interact one with the other, it is reasonable to start by requiring that the laws describing the sub-domain of a_1 to be in a separate module than those describing the domain of a_2. Lets suppose that is the case, i.e., we have $\mathcal{D}_1 = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T}_1 \rangle$ and $\mathcal{D}_2 = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T}_2 \rangle$, with

$$\mathcal{T}_1 = \left\{ \begin{array}{l} p \rightarrow [a_1]\bot, \\ p \rightarrow \langle a_1 \rangle \top \end{array} \right\}, \; \mathcal{T}_2 = \{\langle a_2 \rangle \top\}$$

(Note that the description is still consistent.) We point out that such a modularization does not satisfy the principle of modularity above: there is a formula, viz. $[a_2]\neg p$ that is entailed by the whole description but is not entailed by the module $\mathcal{D}_2$ alone. This means our decomposition of $\mathcal{D}$ in $\mathcal{D}_1$ and $\mathcal{D}_2$ is not good. But where is the problem? We said that a_1 and a_2 play no role together. So why $\mathcal{D}_2$ alone is not enough to derive all conclusions in the domain of a_2? Because there is an *implicit logical interaction* between laws for a_1 and a_2 that cannot be avoided. Zooming in inside $\mathcal{T}_1$, we see that it entails $\neg p$, i.e., $\neg p$ is a static law (hence, valid in every possible state of the world), and, because the same happens in $\mathcal{T}$, we have $\mathcal{T} \models_{\mathsf{PDL}} \neg p$ and then $\mathcal{T} \models_{\mathsf{PDL}} [a_2]\neg p$. Such a global implicit entailment "gets lost" when we decompose the description, and that

[3] *Ex falso sequitur quodlibet*, the law of classical logic according to which "anything follows from a contradiction."

is what makes the result to violate modularity. In order to overcome the problem, we should rather join both modules. However, this gives exactly $\mathcal{D}$ as result!

Here we argue that formulas in reasoning about actions are so coupled, so related that it is infeasible to have at once local completeness and intelligibility with scalability. For applications of real interest, modules have to be so huge that we will find inside the module the original problem about the initial description: it is big, difficult to understand and whose pretended independence from the other modules falls down if a change in some other module forces an implicit law.

To summarize, either sub-domains are put together, giving us huge modules, with lots of different types of formulas mixed and whose intelligibility is doubtful, or we redefine modularity, probably relaxing it, to allow the (natural and unavoidable) coupling among different formulas. Here we chose the second way and that is the issue we henceforth address.

Chapter 4

where we show how we cut the grass in our garden

The Modularity's New Clothes

Take what you can use and let the rest go by.

— Ken Kesey

In this chapter, we make a step further through the notion of modularity of an action theory and analyze some of its properties. We propose a way to overcome the problem of implicit laws that we saw in the last chapter. For the sake of simplicity, we suppose that no solution to the frame problem is given. We propose algorithms to check whether a given action theory has implicit laws and that also catch them. Completeness, correctness and termination results are demonstrated.

4.1 A Natural Decomposition

We start by observing that it is often the case that a set of axioms $\mathcal{T}$ containing multiple modalities $a_1, a_2, \ldots$ can be naturally partitioned into a union of theories $\mathcal{T}^{\emptyset} \cup \mathcal{T}^{a_1} \cup \mathcal{T}^{a_2} \cup \ldots$ such that $\mathcal{T}^{\emptyset}$ contains no modal operators, and the only modality appearing in each $\mathcal{T}^{a_i}$ is a_i.

For example, consider an action theory $\mathcal{D}_{marriage} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$ such that:

$$\mathcal{T} = \left\{ \begin{array}{c} \neg(married \wedge bachelor), \\ \neg married \rightarrow \langle marry \rangle \top, \\ [marry] married \end{array} \right\}$$

We can see such a theory as composed of two modules, one for expressing the *dynamic* part of the theory, and another one to formalize the static constraints of the domain.

The module

$$\mathcal{D}_{marriage1} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \left\{ \begin{array}{c} \neg married \rightarrow \langle marry \rangle \top, \\ [marry] married \end{array} \right\} \rangle$$

formalizes the behavior of the action of getting married, in this case the precondition for executing *marry* (viz. $\neg married$) and the effect that obtains after its execution (viz. *married*). The module

$$\mathcal{D}_{marriage2} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \{\neg(married \wedge bachelor)\} \rangle$$

formalizes the static law according to which it is not possible to be married and bachelor at the same time.

Let the underlying multimodal logic be independently axiomatized (cf. Section 2.1), and suppose we want to know whether $\mathcal{D} \models \Phi$, i.e., whether a formula Φ follows from the action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$. Then it is natural to expect that we only have to consider those elements of $\mathcal{T}$ which concern the modal operators occurring in Φ. For instance, the proof of some consequences of action a_1 should not involve laws for other actions a_2. Note that this is not the case if the logic is not independently axiomatized and there are interaction axioms such as $[a_1]\Phi \rightarrow [a_2]\Phi$.

Here we propose a modality-based decomposition of an action theory $\mathcal{D}$.

Let $act(\Phi)$ return the set of modal operators (actions) occurring in formula Φ, and, for given $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$, let $act(\mathcal{T}) = \bigcup_{\Phi \in \mathcal{T}} act(\Phi)$. For instance, $act([a_1](p_1 \rightarrow [a_2]p_2)) = \{a_1, a_2\}$. For given $a \in \mathfrak{Act}$, we define

$$\mathcal{T}^a = \{\Phi \in \mathcal{T} \ : \ act(\Phi) = \{a\}\}$$

For formulas with no modality, we define

$$\mathcal{T}^{\emptyset} = \{\Phi \in \mathcal{T} \ : \ act(\Phi) = \emptyset\}$$

For example, if

$$\mathcal{T} = \left\{ \begin{array}{c} \neg(married \wedge bachelor), \\ \neg married \rightarrow \langle marry \rangle \top, [marry] married, \\ married \rightarrow \langle divorce \rangle \top, [divorce] \neg married \end{array} \right\}$$

then

$$\mathcal{T}^{divorce} = \left\{ \begin{array}{c} married \rightarrow \langle divorce \rangle \top, \\ [divorce] \neg married \end{array} \right\}$$

and

$$\mathcal{T}^{\emptyset} = \{\neg(married \wedge bachelor)\}$$

We henceforth make the following hypothesis:

$$\{\mathcal{T}^{\emptyset}\} \cup \{\mathcal{T}^{a_i} \ : \ a_i \in \mathfrak{Act}\} \text{ partitions}^1 \ \mathcal{T} \tag{H}$$

We thus exclude $\mathcal{T}^{a_i}$ containing more than one modal operator.

Given this, we are now able to formally define modularity of a theory.

4.2 Modularity

We are interested in the following principle of modularity:

Definition 4.1 (Modularity)
An action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$ *is* modular *if and only if for every formula* Φ*,*

$$\mathcal{D} \models \Phi \text{ implies } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T}^{\emptyset} \cup \mathcal{T}^{act(\Phi)} \rangle \models \Phi.$$

Our notion of modularity means that when investigating whether Φ is a consequence of $\mathcal{D}$, the only formulas of $\mathcal{D}$ that are relevant are those whose modal operators occur in Φ and the classical formulas in $\mathcal{T}^{\emptyset}$.

This is reminiscent of interpolation [21], which more or less[2] says:

Definition 4.2 (Interpolation property)
An action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$ *has the* interpolation property *if and only if for every formula* Φ*, if* $\mathcal{D} \models \Phi$*, then there is a module* $\mathcal{D}_\Phi = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T}_\Phi \rangle$ *such that*

- $act(\mathcal{T}_\Phi) \subseteq act(\mathcal{T}) \cap act(\Phi)$*;*
- $\mathcal{D} \models \Phi'$ *for every* $\Phi' \in \mathcal{T}_\Phi$*; and*
- $\mathcal{D}_\Phi \models \Phi$.

[1] $\{\mathcal{T}^{\emptyset}\} \cup \{\mathcal{T}^{a_i} \ : \ a_i \in \mathfrak{Act}\}$ partitions $\mathcal{T}$ if and only if $\mathcal{T} = \mathcal{T}^{\emptyset} \cup \bigcup_{a_i \in \mathfrak{Act}} \mathcal{T}^{a_i}$, and $\mathcal{T}^{\emptyset} \cap \mathcal{T}^{a_i} = \emptyset$, and $\mathcal{T}^{a_i} \cap \mathcal{T}^{a_j} = \emptyset$, if $a_i \neq a_j$. Note that $\mathcal{T}^{\emptyset}$ and each $\mathcal{T}^{a_i}$ might be empty.

[2] We here present a version in terms of global consequence, as opposed to local consequence or material implication versions that can be found in the literature [69, 70]. We were unable to find such global versions in the literature.

Our definition of modularity is a strengthening of interpolation because it requires $\mathcal{T}_\Phi$ to be a subset of $\mathcal{T}$. Properties similar to interpolation for reasoning about actions in PDL have also been investigated in [120].

Contrary to interpolation, modularity does not generally hold. Clearly if the Hypothesis (H) is not satisfied, then modularity fails. To witness, consider $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$ such that

$$\mathcal{T} = \{p_1 \rightarrow [a_1][a_2]p_2, [a_1][a_2]p_2 \rightarrow p_3\}$$

Then $\mathcal{D} \models p_1 \rightarrow p_3$, but $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T}^\emptyset \cup \mathcal{T}^{act(p_1 \rightarrow p_3)} \rangle \not\models p_1 \rightarrow p_3$.

Nevertheless, even under our hypothesis, modularity may fail to hold. For example, let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$ be such that

$$\mathcal{T} = \{\neg p \rightarrow [a]\bot, \neg p \rightarrow \langle a \rangle \top\}$$

Then $\mathcal{T}^\emptyset = \emptyset$, and $\mathcal{T}^a = \mathcal{T}$. Now $\mathcal{D} \models p$, but clearly $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T}^\emptyset \cup \mathcal{T}^{act(p)} \rangle \not\models p$.

How can we know whether a given action theory $\mathcal{D}$ is modular? The following criterion is simpler:

Definition 4.3 (Propositional modularity)
An action theory $\mathcal{D}$ is propositionally modular *if and only if for every propositional formula φ,*

$$\mathcal{D} \models \varphi \textit{ implies } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T}^\emptyset \rangle \models \varphi$$

And that suffices to guarantee modularity:

Theorem 4.1
Let the underlying logic be a fusion, and let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$ be such that $\mathcal{T}$ is partitioned. If $\mathcal{D}$ is propositionally modular, then $\mathcal{D}$ is modular.

Proof:
See Appendix A. ∎

In the rest of the chapter, we investigate how it can be automatically checked whether a given action theory $\mathcal{D}$ is modular or not, and how to make it modular, if needed.

4.3 Deciding Modularity

How can we check whether a given action theory $\mathcal{D}$ is modular? Following Theorem 4.1, it is enough to check for propositional modularity.

Definition 4.4 (Implicit static law)
$\varphi \in \mathfrak{Fml}$ *is an* implicit static law *of an action theory* $\mathcal{D}$ *if and only if* $\mathcal{D} \models \varphi$ *and* $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{T}^{\emptyset} \rangle \not\models \varphi$.

Let $\mathcal{D}_{trans} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$ formalize a transaction domain such that:

$$\mathcal{T} = \left\{ \begin{array}{c} \neg adult \rightarrow \neg obligedPay, [order]obligedPay, \\ \neg adult \rightarrow [order]\neg adult, \langle order \rangle \top \end{array} \right\}$$

Observe that by the fact that $\mathcal{D}_{trans} \models \neg adult \rightarrow [order]\bot$, we have $\mathcal{D}_{trans} \models adult$. But $\mathcal{T}^{\emptyset} \not\models_{\overline{\mathsf{CPL}}} adult$, hence *adult* is an example of an implicit static law . Moreover, $\mathcal{D}_{trans}$ is also an example of an action theory that is not modular in our sense.

Theorem 4.1 tells us that an action theory is modular if and only if it has no implicit static law. Hence, checking the existence of such laws provides us a way to decide modularity of a given action theory. Assuming the theory component $\mathcal{T}$ of an action theory is finite, with Algorithm 4.1 below we can check whether an action theory has such implicit laws. The idea is as follows: for each pair of laws $\varphi_1 \rightarrow \langle a \rangle \top$ and $\varphi_2 \rightarrow [a]\bot$ in $\mathcal{T}$, if $\varphi_1 \wedge \varphi_2$ is satisfiable and $\mathcal{T}^{\emptyset} \not\models_{\overline{\mathsf{CPL}}} \neg(\varphi_1 \wedge \varphi_2)$, mark $\neg(\varphi_1 \wedge \varphi_2)$ as an implicit static law.

Algorithm 4.1 Deciding existence of implicit static laws

input: $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \rangle$
output: a set of implicit static laws $\mathcal{S}_{imp}$
  $\mathcal{S}_{imp} := \emptyset$
  for all $a \in act(\mathcal{T})$ **do**
    for all $\varphi' \rightarrow \langle a \rangle \top \in \mathcal{T}$ **do**
      for all $\{\varphi_1 \rightarrow [a]\psi_1, \ldots, \varphi_n \rightarrow [a]\psi_n\} \subseteq \mathcal{T}^a$ **do**
        if $\mathcal{T}^{\emptyset} \cup \{\varphi', \varphi_1, \ldots, \varphi_n\} \not\models_{\overline{\mathsf{CPL}}} \bot$ **and** $\mathcal{T}^{\emptyset} \cup \{\psi_1, \ldots, \psi_n\} \models_{\overline{\mathsf{CPL}}} \bot$ **then**
          $\mathcal{S}_{imp} := \mathcal{S}_{imp} \cup \{\neg(\varphi' \wedge \varphi_1 \wedge \ldots \wedge \varphi_n)\}$

Theorem 4.2 (Decidability)
Algorithm 4.1 terminates.

Proof:
Straightforward from finiteness of $\mathcal{T}$. ∎

Theorem 4.3 (Soundness)
Let $\mathcal{S}_{imp}$ be the output of Algorithm 4.1 on input $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$. Then every $\varphi \in \mathcal{S}_{imp}$ is an implicit static law of $\mathcal{D}$.

Proof:
Let $\varphi \in \mathfrak{Fml}$ be such that $\varphi \in \mathcal{S}_{imp}$ and $\mathcal{D} \models \varphi$. φ is of the form $\neg(\varphi' \wedge \varphi_1 \wedge \ldots \wedge \varphi_n)$, for some $\varphi', \varphi_1, \ldots, \varphi_n$, and $\mathcal{T}^{\emptyset} \cup \{\varphi' \wedge \varphi_1 \wedge \ldots \wedge \varphi_n\} \not\models_{\mathsf{CPL}} \bot$ is the case. Hence, $\mathcal{T}^{\emptyset} \cup \{\neg\varphi\} \not\models_{\mathsf{CPL}} \bot$, which means that $\mathcal{T}^{\emptyset} \not\models_{\mathsf{CPL}} \varphi$. Therefore φ is an implicit static law. ■

Remark 4.1 The converse of Theorem 4.3 does not hold: consider the quite simple action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$ such that

$$\mathcal{T} = \left\{ \begin{array}{c} \neg p_n, \langle a \rangle \top, \\ p_{i-1} \rightarrow [a] p_i,\ 1 \leq i \leq n \end{array} \right\}$$

Thus, $\mathcal{D} \models \neg p_i$, for $0 \leq i \leq n$, but running Algorithm 4.1 returns only $\mathcal{S}_{imp} = \{\neg p_{n-1}\}$.

This suggests that it is necessary to *iterate* the algorithm in order to find all implicit static laws. We shall do this in the sequel, and now just observe that:

Theorem 4.4
An action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$ is modular if and only if $\mathcal{S}_{imp} = \emptyset$.

Proof:
See Appendix A. ■

Considering the action theory in Remark 4.1, we see that running Algorithm 4.1 on $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \cup \{\neg p_{n-1}\} \rangle$ gives us $\mathcal{S}_{imp} = \{\neg p_{n-2}\}$. This means some of the implicit static laws may be needed in order to derive others. Hence, Algorithm 4.1 should be iterated to get $\mathcal{D}$ modular. This is achieved with Algorithm 4.2, which iteratively feeds the set of static laws considered into the **if**-test of Algorithm 4.1.

Algorithm 4.2 Finding all implicit static laws

input: $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$
output: $\mathcal{S}_{imp^*}$, the set of all implicit static laws of $\mathcal{D}$
  $\mathcal{S}_{imp^*} := \emptyset$
  repeat
    $\mathcal{S}_{imp} := find\_imp\_stat(\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \cup \mathcal{S}_{imp^*} \rangle)$ {a call to Algorithm 4.1}
    $\mathcal{S}_{imp^*} := \mathcal{S}_{imp^*} \cup \mathcal{S}_{imp}$
  until $\mathcal{S}_{imp} = \emptyset$

Theorem 4.5 (Decidability)
Algorithm 4.2 terminates.

Proof:
First, for given a the set of candidates to be an implicit static law is

$$\{\neg(\varphi \wedge \bigwedge_{\varphi_i \to [a]\psi_i \in \widehat{\mathcal{T}^a}} \varphi_i) \,:\, \varphi \to \langle a\rangle\top \in \mathcal{T}^a \text{ and } \widehat{\mathcal{T}^a} \subseteq \mathcal{T}^a\}$$

This set is finite.

In each step, either the algorithm ends because $\mathcal{S}_{imp} = \emptyset$, or at least one of the candidates is put into $\mathcal{S}_{imp}$ (by a call to Algorithm 4.1, which terminates). Such a candidate is not going to be put into $\mathcal{S}_{imp}$ in future steps, because once added to $\mathcal{S}_{imp^*}$, it will be in the set of laws of all subsequent calls to Algorithm 4.1, falsifying its respective **if**-test for such a candidate. Hence the **repeat**-loop is bounded by the number of candidates, and therefore Algorithm 4.2 terminates. ■

Theorem 4.6
Let $\mathcal{S}_{imp^*}$ be the output of Algorithm 4.2 on input $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T}\rangle$. Then

1. *$\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \cup \mathcal{S}_{imp^*}\rangle$ is modular.*
2. *$\mathcal{D} \models \bigwedge \mathcal{S}_{imp^*}$.*

Proof:
Item 1. is straightforward from the termination of Algorithm 4.2 and Theorem 4.4. Item 2. follows from the fact that by the **if**-test in Algorithm 4.1, the only formulas that are put in $\mathcal{S}_{imp^*}$ at each execution of the loop are exactly those that are implicit static laws of the original theory. ■

Corollary 4.1
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T}\rangle$. For all $\varphi \in \mathfrak{Fml}$, $\mathcal{D} \models \varphi$ if and only if $\mathcal{T}^{\emptyset} \cup \mathcal{S}_{imp^*} \models_{\overline{\mathsf{CPL}}} \varphi$.

Proof:
For the left-to-right direction, let $\varphi \in \mathfrak{Fml}$ be such that $\mathcal{D} \models \varphi$. Then $\mathcal{T} \models_{\overline{\mathsf{PDL}}} \varphi$, and hence $\mathcal{T} \cup \mathcal{S}_{imp^*} \models_{\overline{\mathsf{PDL}}} \varphi$, by monotonicity. By Theorem 4.6-1., $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{T} \cup \mathcal{S}_{imp^*}\rangle$ is modular, hence $\mathcal{T}^{\emptyset} \cup \mathcal{S}_{imp^*} \models_{\overline{\mathsf{CPL}}} \varphi$.

The right-to-left direction is straightforward by Theorem 4.6-2. ■

This establishes that Algorithm 4.2 finds all implicit static laws of an action theory $\mathcal{D}$. Adding such laws to the theory component $\mathcal{T}$ of $\mathcal{D}$ guarantees, hence, modularity of $\mathcal{D}$.

4.4 What about the Frame Problem?

As the reader might have already expected, the formalism of PDL alone does not solve the frame problem [90]. For instance, if $\mathcal{D}_{wts}$ describes our shooting domain (cf. Section 2.3), then

$$\mathcal{D}_{wts} \not\models hasGun \rightarrow [load]hasGun.$$

The reason is that there are some *anomalous models* in which *hasGun* is not preserved after the execution of action *load*. The model in Figure 2.3 is an example of an anomalous model for such a scenario. Figure 4.1 highlights one of its problematic transitions.

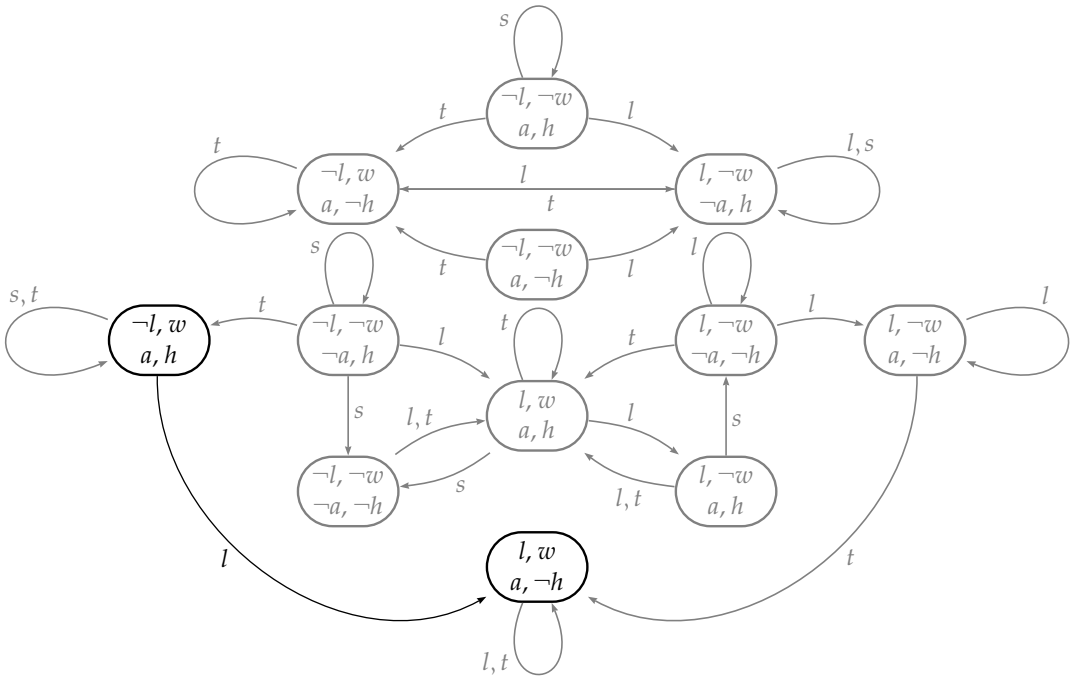

Figure 4.1: Anomalous model in the Walking Turkey Scenario.

The formula $hasGun \rightarrow [load]hasGun$ is an example of a frame axiom. Following the tradition in the reasoning about actions community, we do not want to state such a kind of axioms in the action theory. In order to satisfy this requirement, given an action theory $\mathcal{D}$, we need a consequence relation powerful enough to deal with the frame problem. This means that the deductive power of PDL has to be augmented in order to ensure that the non-effects of actions follow from the theory. We here opt for the dependence-based approach presented in [14]. There, meta-logical information, given in the form of a dependence relation, is added to PDL.

Definition 4.5 (Dependence relation [14])
A dependence relation *is a binary relation* $\rightsquigarrow \subseteq \mathfrak{Act} \times \mathfrak{Lit}$.

The expression $a \rightsquigarrow \ell$ denotes that the execution of action *a may* make the literal ℓ true. In our example, we have

$$\rightsquigarrow = \left\{ \begin{array}{c} \langle shoot, \neg loaded\rangle, \langle shoot, \neg alive\rangle, \\ \langle shoot, \neg walking\rangle, \langle tease, walking\rangle \end{array} \right\},$$

which means that action *shoot* may make the literals $\neg$*loaded*, $\neg$*alive* and $\neg$*walking* true, and action *tease* may make *walking* true.

Semantically, the dependence-based approach relies on the explanation closure assumption [102], and its solution to the frame problem consists in a kind of negation as failure: because $\langle load, \neg hasGun\rangle \notin \rightsquigarrow$, we have $load \not\rightsquigarrow \neg hasGun$, i.e., $\neg hasGun$ is never caused by *load*. Thus, in a context where *hasGun* is true, after every execution of *load*, *hasGun* still remains true. We also have $tease \not\rightsquigarrow alive$ and $tease \not\rightsquigarrow \neg alive$. The meaning of all these independences is that the frame axioms $hasGun \rightarrow [load]hasGun$, $\neg alive \rightarrow [tease]\neg alive$ and $alive \rightarrow [tease]alive$ hold.

We assume that $\rightsquigarrow$ is finite.

A dependence relation $\rightsquigarrow$ defines a class of possible worlds models:

Definition 4.6 ($\rightsquigarrow$ truth conditions)
A PDL*-model* $\mathscr{M} = \langle W, R\rangle$ *is a* $\rightsquigarrow$-model *if and only if whenever* wR_aw' *then:*

- *if* $a \not\rightsquigarrow p$, *then* $\models^{\mathscr{M}}_{w} p$ *implies* $\models^{\mathscr{M}}_{w'} p$; *and*
- *if* $a \not\rightsquigarrow \neg p$, *then* $\not\models^{\mathscr{M}}_{w} p$ *implies* $\not\models^{\mathscr{M}}_{w'} p$.

Figure 4.2 depicts the dependence-based condition on models.

Given a $\rightsquigarrow$-model $\mathscr{M}$, Φ and $\mathcal{T}$, $\models^{\mathscr{M}} \Phi$ and $\models^{\mathscr{M}} \mathcal{T}$ are defined as in Definition 2.3.

Definition 4.7 ($\rightsquigarrow$-based logical consequence)
A formula Φ *is a* $\rightsquigarrow$-based consequence *of the set of global axioms* $\mathcal{T}$ *in the class of all* $\rightsquigarrow$*-models (noted* $\mathcal{T} \models_{\rightsquigarrow} \Phi$*) if and only if for every* $\rightsquigarrow$*-model* $\mathscr{M}$, *if* $\models^{\mathscr{M}} \mathcal{T}$, *then* $\models^{\mathscr{M}} \Phi$.

Thus, if in our example we replace in $\mathcal{D}_{wts}$ the consequence relation $\models_{\rightsquigarrow}$, with its associated dependence relation above, for $\models_{\mathsf{PDL}}$, it holds:

$$\mathcal{D}_{wts} \models hasGun \rightarrow [load]hasGun.$$

In this way, the dependence-based approach solves the frame problem.

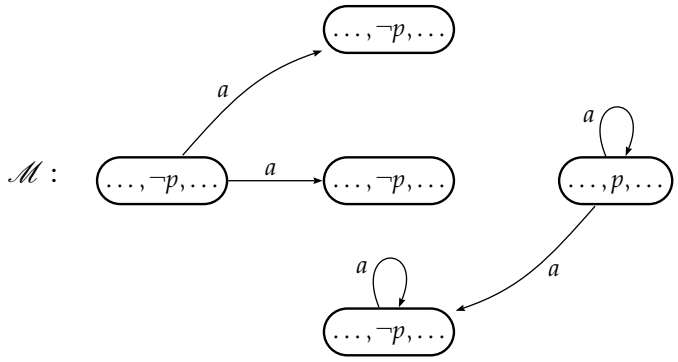

Figure 4.2: Dependence-based condition: preservation of literal $\neg p$ under hypothesis $a \not\rightsquigarrow p$.

Henceforth we consider $\models_{\not\rightsquigarrow}$ as the consequence relation component of our PDL domain descriptions.

Definition 4.8 (Action theory model)
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\not\rightsquigarrow}, \mathcal{T} \rangle$ be a domain description. A PDL*-model $\mathscr{M}$ is a* model of $\mathcal{D}$ *if and only if $\mathscr{M}$ is a $\rightsquigarrow$-model and $\models^{\mathscr{M}} \mathcal{T}$.*

All the definitions we have given in this chapter shall then be adapted to fit together with our new consequence relation. We will develop this issue later on in this work. Before doing that, in the next two chapters we investigate interesting properties that justify our choice for the dependence-based approach.

Chapter 5

where we show we can do as simple as others

Recasting Reiter's Solution

Almost all absurdity of conduct arises from the imitation of those whom we cannot resemble.

— Samuel Johnson

In this chapter, we propose an encoding of Reiter's Situation Calculus solution to the frame problem into the framework of our multimodal logic of actions. In particular, we show that with the dependence-based solution to the frame problem we achieve that without quantification, and present the modal counterpart of the regression technique. This gives us a theorem proving method for a relevant fragment of our dynamic logic.

5.1 Deterministic PDL with Quantification and Equality

In the reasoning about actions field, most approaches use the Situation Calculus formalism [90]. Among those, Reiter's [99] has turned out to be most fruitful. His basic formalism is restricted to deterministic actions without static laws. In order to solve the frame problem, he makes use of so-called Successor State Axioms (SSAs). The latter enable regression [99], which has interesting computational properties.

The Situation Calculus is a dialect of predicate logic, having situations and actions as objects, and where actions are viewed as mappings on the set of situations. At first glance, this is very close to possible worlds semantics for deterministic PDL [49]. But the precise relation between Reiter's approach and dynamic logic is not as obvious as that. One of the reasons why his formalism cannot be translated straightforwardly

into modal logics of action such as PDL is that the Situation Calculus allows quantifying over actions. Worse, such quantifications are central to Reiter's approach.

In [24] there has been presented a technique to translate Reiter's framework into dynamic logic. In this chapter we present a different approach. We solve the problem using the dependence-based extension to PDL that we saw in the previous chapter. Having such a result provides some degree of optimization in doing inference tasks for some important classes of problems in the area.

In this chapter, we will concentrate only on deterministic PDL, i.e., the logic we have defined in Chapter 2 restricted to the case where each R_a is deterministic: for each action a and each world w, there is at most one world w' such that wR_aw'. Moreover, we here slightly extend such a logic in order to allow for quantification over actions and the equality predicate. This will serve as the basis for developing the ideas in this chapter.

We here will use $\vec{a}$ as a meta-variable ranging over action constants and variables. Here Φ will also denote complex formulas possibly involving quantification and equality between actions.

The nonstandard feature of the logic we are going to use here is that we allow for *quantification over actions*, and for *equality between actions*. Hence, in this version of dynamic logic, we allow for formulas of the form $\forall a.\Phi$, with Φ a complex formula. In the Yale shooting scenario (YSS) [47], one can e.g. write

$$\forall a.(alive \wedge \neg[a]alive \rightarrow (a = shoot \wedge hasGun \wedge loaded)).$$

This is an *explanation closure axiom* [102] expressing that the only way to make *alive* false is by the *shoot* action under preconditions *hasGun* and *loaded*.

We call our version of deterministic PDL with quantification and equality DPDL$^+$.

Once added these features to deterministic PDL, it remains to redefine what its models are.

Definition 5.1 (DPDL$^+$-model)
A DPDL$^+$-model *is a triple* $\mathscr{M} = \langle W, R, I\rangle$ *where* W *and* R *are as in Definition 2.1, and* I *is an* interpretation function *mapping propositional constants to subsets of* W*, and action constants and variables to elements of* R*.*

We will sometimes write $w' \in (I(\vec{a}))(w)$ instead of $wI(\vec{a})w'$, and similarly for variables a.

Definition 5.2 (Interpretation agreement)
Let I and I' be interpretations. I agrees with I' except possibly on a *if and only if*

- $I(p) = I'(p)$, *for every propositional constant* p;
- $I(\vec{a}) = I'(\vec{a})$, *for every action constant* $\vec{a}$; *and*
- $I(a') = I'(a')$, *for every action variable* a' *different from* a.

For a DPDL$^+$-model $\mathscr{M} = \langle W, R, I\rangle$, $\models^{\mathscr{M}}_{w} \forall a.\Phi$ if and only if for every I' such that I agrees with I' except possibly on a, $\models^{\langle W,R,I'\rangle}_{w} \Phi$. $\models^{\mathscr{M}}_{w} [\vec{a}]\Phi$ if and only if for every $w' \in (I(\vec{a}))(w)$, $\models^{\mathscr{M}}_{w'} \Phi$. $\models^{\mathscr{M}}_{w} [a]\Phi$ if and only if for every $w' \in (I(a))(w)$, $\models^{\mathscr{M}}_{w'} \Phi$. The other truth conditions, truth in a model and logical consequence are as defined in Section 2.1.

Actions being *deterministic*, i.e., $(I(\vec{a}))(w)$ is either a singleton or empty, we have that for every action constant $\vec{a}$ and every formula Φ

$$\models_{\mathsf{DPDL}^+} \langle\vec{a}\rangle\Phi \rightarrow [\vec{a}]\Phi \tag{5.1}$$

If all actions are deterministic, then every formula without quantification can be brought into a normal form where there are neither conjunctions nor disjunctions in the scope of modal operators. Apart from classical equivalences, this uses the following ones from the left to the right:

$$\models_{\mathsf{DPDL}^+} [\vec{a}](\Phi \wedge \Phi') \leftrightarrow ([\vec{a}]\Phi \wedge [\vec{a}]\Phi') \tag{5.2}$$

$$\models_{\mathsf{DPDL}^+} [\vec{a}](\Phi \vee \Phi') \leftrightarrow ([\vec{a}]\Phi \vee [\vec{a}]\Phi') \tag{5.3}$$

In the next section, we introduce the basic hypotheses concerning the knowledge we have about actions.

5.2 Describing Actions Like Reiter

In describing an action theory, it is more or less explicitly supposed that the following pieces of information are given. (Some assumptions of complete information are made about them.)

For each action constant $\vec{a}$, there is a classical formula $Poss(\vec{a})$ describing the action precondition of $\vec{a}$, i.e., the condition under which $\vec{a}$ can be executed. For example, $Poss(shoot) = hasGun$, and $Poss(strangle) = \top$.

It is supposed that action preconditions are *complete*: $\vec{a}$ is executable if and only if $Poss(\vec{a})$ is true. In terms of dynamic logic, completeness of action preconditions means that for every $\vec{a} \in \mathfrak{Act}$, we have a global axiom:

$$Poss(\vec{a}) \leftrightarrow \neg[\vec{a}]\bot \tag{5.4}$$

For each propositional constant p, there are two *finite* sets of action constants $causes^+(p)$ and $causes^-(p)$, describing, respectively, the positive and negative causes of p. The set $causes^+(p)$ contains the actions in $\mathfrak{Act}$ which in some circumstances might cause p to become true, while $causes^-(p)$ contains those actions that may cause p false. For example, $causes^+(alive) = \emptyset$ (no action makes an agent alive), $causes^-(alive) = \{shoot, strangle\}$, and $causes^-(loaded) = \{shoot\}$.[1]

It is also supposed that $causes^+(p)$ and $causes^-(p)$ are small, in the sense that $causes^+(p)$ and $causes^-(p)$ are much smaller than $\mathfrak{Act}$.

Moreover, we suppose that these two sets are *complete*: whenever $\vec{a} \notin causes^+(p)$, then the execution of $\vec{a}$ can never make p true. In terms of dynamic logic, causal completeness means that we have a global axiom $\neg p \rightarrow [\vec{a}]\neg p$ in that case. Similarly, for every $\vec{a}'$ such that $\vec{a}' \notin causes^-(p)$ we have a global axiom $p \rightarrow [\vec{a}']p$. These are frame axioms. In our example, as $strangle \notin causes^-(loaded)$, we have $loaded \rightarrow [strangle]loaded$. This corresponds to the explanation closure assumption [102, 103].

For all propositional constant $p \in \mathfrak{Prop}$ and every action constant $\vec{a} \in causes^+(p)$, there is a classical formula $Cond^+(\vec{a}, p)$ describing the *positive effect precondition* of action $\vec{a}$. As an example, $Cond^+(toggle, up) = \neg up$, and $Cond^+(load, loaded) = \top$. Similarly, for every $\vec{a} \in causes^-(p)$, there is a $Cond^-(\vec{a}, p)$ describing its *negative effect precondition*. For example, $Cond^-(strangle, alive) = \top$, and $Cond^-(shoot, alive) = loaded$.[2]

It is supposed that effect preconditions are *complete*: in situations where the formula $Cond^+(\vec{a}, p)$ does not hold, the execution of $\vec{a}$ can never make p true. Symmetrically, when $Cond^-(\vec{a}, p)$ does not hold, then the execution of $\vec{a}$ can never make p false.

In terms of dynamic logic, to every effect precondition $Cond^+(a, p)$, one can associate a global axiom $Cond^+(\vec{a}, p) \rightarrow [\vec{a}]p$, and to every effect precondition $Cond^-(a, p)$, one can associate a global axiom $Cond^-(a, p) \rightarrow [\vec{a}]\neg p$. As an example, we have the formula $loaded \rightarrow [shoot]\neg alive$.

Completeness of effect preconditions means that we moreover have a global axiom $(\neg Cond^+(\vec{a}, p) \wedge \neg p) \rightarrow [\vec{a}]\neg p$ for every $\vec{a} \in causes^+(p)$. Symmetrically, for every $\vec{a}'$

[1] In Reiter's presentation, these functions can be retrieved from his functions γ^+ and γ^- [100].

[2] These functions correspond to Reiter's γ^+ and γ^-.

such that $\vec{a}' \in causes^{-}(p)$, we have a global axiom $(\neg Cond^{-}(\vec{a}', p) \wedge p) \rightarrow [\vec{a}']p$. This expresses in dynamic logic what Reiter calls the application of Clark completion [18]. For example, we have $(\neg loaded \wedge alive) \rightarrow [shoot]alive$. In [14], axioms of this form are called *conditional frame axioms*. There they are needed to complete the dependence relation so that we can capture context-dependent effects of actions.

The three pieces of information together with the completeness assumptions guarantee that the possible world resulting from the execution of action $\vec{a}$ in a possible world w is completely determined: for every model $\mathscr{M} = \langle W, R, I\rangle$ and every world $w \in W$, if $\not\models^{\mathscr{M}}_{w} Poss(\vec{a})$, then $(I(\vec{a}))(w) = \emptyset$. Else, the truth value of every p in every w' accessible from w via $I(\vec{a})$ is as follows. Suppose w.l.o.g. that $\models^{\mathscr{M}}_{w} p$. Then:

- if $\vec{a} \notin causes^{-}(p)$, then $\models^{\mathscr{M}}_{w'} p$;
- if $\vec{a} \in causes^{-}(p)$ and $\not\models^{\mathscr{M}}_{w} Cond^{-}(\vec{a}, p)$, then $\models^{\mathscr{M}}_{w'} p$; and
- if $\vec{a} \in causes^{-}(p)$ and $\models^{\mathscr{M}}_{w} Cond^{-}(\vec{a}, p)$, then $\not\models^{\mathscr{M}}_{w'} p$.

As all truth values are thus determined, it follows that the set of worlds accessible via $I(\vec{a})$ is either empty, or it can be considered to be a singleton. This fits with the assumption that all actions are deterministic.

As we have noted, the action preconditions and effect preconditions appear explicitly in Reiter's formalization, while the sets of possible causes $causes^{+}(p)$ and $causes^{-}(p)$ only appear implicitly there.

Note that in Reiter's Situation Calculus it is supposed that actions always lead to some state: even in states where the agent has no gun in his hands, the state resulting from the execution of *shoot* exists. The technical reason is that just as every function in predicate logic, his successor function $do(.)$ is total. This means that the logic of each action operator $[\vec{a}]$ should be KD [16]. We have nevertheless decided to follow the dynamic logic tradition and suppose that the set of worlds accessible via some action a might be empty. Therefore the logic of each $[\vec{a}]$ is just K.

In fact, inexecutability of the action *shoot* is expressed in Situation Calculus by stating $Poss(shoot) \leftrightarrow hasGun$, where $Poss(shoot)$ is a particular propositional constant. In our formulation, $Poss(.)$ is a function associating a classical formula to every action $\vec{a}$. $Poss(\vec{a})$ can be seen as an abbreviation, such as $Poss(shoot) = hasGun$. Given a domain description in Reiter's style, we obtain a description in our style if we

- Define our $Poss(\vec{a})$-function from Reiter's preconditions $Poss(\vec{a}) \leftrightarrow \varphi$; and

- Replace Reiter's constants $Poss(\vec{a})$ by our $\langle\vec{a}\rangle\top$.

The other way round, our version can be translated to Reiter's by

- Defining his preconditions $Poss(\vec{a}) \leftrightarrow \varphi$ from our $Poss(\vec{a})$-function; and
- Recursively replacing $[\vec{a}]\varphi$ by $Poss(\vec{a}) \rightarrow [\vec{a}]\varphi$.

Observe that the latter is nothing but the well-known translation from modal logic K to KD [93, 94].

All this sounds as if action theories could be described in DPDL^+ in a satisfactory manner, but, in such a framework, we have not solved the frame problem yet: as by hypothesis $causes^+(p)$ and $causes^-(p)$ are small, it follows that the size of the set of frame axioms that we have to state is close to $card(\mathfrak{Prop}) \times card(\mathfrak{Act})$. This is usually considered to be too big, and a central element in the research program of the reasoning about actions community was to design mechanisms allowing to infer such frame axioms without stating them explicitly.

There was a 20-years-long debate about semantics and theorem proving methods allowing such inferences. Reiter's proposal seems to have closed the debate at least in what concerns deterministic actions and no static laws. This is going to be presented in the sequel.

5.3 Reiter's Solution to the Frame Problem

Based on a particular class of models, Reiter proposes to incorporate the basic ingredients of action theories that we have presented in the preceding section into what he calls *Successor State Axioms* (SSA) [99]. These are special formulas that, given a state and an action, completely determine the next state.

Reiter requires that all object names in the domain signature are *unique* and that models are *trees*.

Definition 5.3 (Reiter model)
A DPDL^+*-model* $\mathscr{M} = \langle W, R, I\rangle$ *is a* Reiter-model *if and only if* $\langle W, \bigcup_{a\in\mathfrak{Act}} R_a\rangle$ *is a tree, and if* $I(\vec{a}_\mathfrak{i}) = I(\vec{a}_\mathfrak{j})$*, then* $\mathfrak{i} = \mathfrak{j}$.

Figure 5.1 illustrates the tree-like structure of a Reiter model.

Definition 5.4 (Reiter's logical consequence)
A formula Φ *is a* Reiter consequence *of the global axioms* $\mathcal{T}$ *in the class of all Reiter-models (noted* $\mathcal{T} \models_{\mathsf{R}} \Phi$*) if and only if for every Reiter-model* $\mathscr{M}$*, if* $\models^{\mathscr{M}} \mathcal{T}$*, then* $\models^{\mathscr{M}} \Phi$.

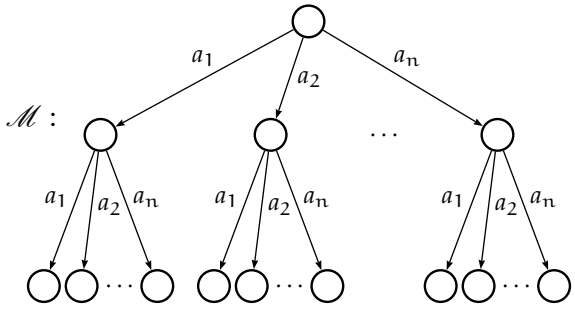

Figure 5.1: Structure of a Reiter-model.

Successor State Axioms

Suppose that all the $Poss(.)$, $causes^+(.)$, $causes^-(.)$, $Cond^+(.)$ and $Cond^-(.)$ are given, and that the completeness assumptions are made. We then can associate with that an action theory $\mathcal{D}_R = \langle \mathcal{L}_{DPDL^+}, \models_R, \mathcal{T}_R \rangle$ from which the relevant frame axioms will follow. According to Reiter's approach, the component $\mathcal{T}_R$ of the description is made up of the following axioms:

- for every $\vec{a} \in \mathfrak{Act}$, there is an executability axiom $Poss(\vec{a}) \leftrightarrow \neg[\vec{a}]\bot$; and
- for every $p \in \mathfrak{Prop}$, if $causes^+(p) = \{a_1, \ldots, a_n\}$ and $causes^-(p) = \{a'_1, \ldots, a'_m\}$, then there is a Successor State Axiom

$$\begin{array}{l}\forall a.([a]p \leftrightarrow \\ (\neg Poss(a) \vee \\ (a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p)))))\end{array}$$

Note that the Successor State Axiom above is well defined because we have supposed that $causes^+(\vec{a})$ and $causes^-(\vec{a})$ are finite.

For the cases where $n = 0$ or $m = 0$, conjunction of the elements of an empty set is identified with $\top$, and disjunction with $\bot$. The latter can be illustrated with our running example, where $causes^+(alive) = \emptyset$. The Successor State Axiom for *alive* is:

$$\begin{array}{l}\forall a.([a]alive \leftrightarrow \\ (\neg Poss(a) \vee \bot \vee (alive \wedge \neg(a = shoot \wedge loaded) \wedge \neg(a = strangle \wedge \top))))\end{array}$$

We abbreviate $reg(a, p)$ the right hand side of such an equivalence. The Successor State Axiom for p therefore has the form $\forall a.([a]p \leftrightarrow reg(a, p))$.

Successor State Axioms can be equivalently stated for negative literals as:

$$\begin{array}{l}\forall a.([a]\neg p \leftrightarrow \\ (\neg Poss(a) \vee (a = a'_1 \wedge Cond^-(a'_1, p)) \vee \ldots \vee (a = a'_m \wedge Cond^-(a_m, p)) \vee \\ (\neg p \wedge \neg(a = a_1 \wedge Cond^+(a_1, p)) \wedge \ldots \wedge \neg(a = a_n \wedge Cond^+(a_n, p)))))\end{array}$$

We abbreviate $reg(a, \neg p)$ the right hand side of this equivalence. For example the Successor State Axiom for $\neg alive$ is:

$$\begin{array}{l}\forall a.([a]\neg alive \leftrightarrow \\ (\neg Poss(a) \vee (a = shoot \wedge loaded) \vee (a = strangle \wedge \top) \vee (\neg alive \wedge \neg\bot)))\end{array}$$

Reiter's original Successor State Axiom [99] is slightly different from ours:

$$\begin{array}{l}\forall a.(Poss(a) \rightarrow ([a]p \leftrightarrow \\ ((a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p)))))\end{array}$$

Our version can be proved to be equivalent to his:

Theorem 5.1

Let $\mathcal{T}$ be the set of global axioms (5.4)–(5.8). Then

$$\begin{array}{ll}\mathcal{T} \models_{\mathsf{DPDL}^+} & (\forall a.(Poss(a) \rightarrow ([a]p \leftrightarrow \\ & ((a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ & (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p)))))))) \\ & \leftrightarrow \\ & (\forall a.([a]p \leftrightarrow \\ & (\neg Poss(a) \vee \\ & (a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ & (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p))))))\end{array}$$

Proof:

See Appendix B. ∎

In [100], Reiter excluded the precondition $Poss(a)$ from SSAs, and then just writes

$$\begin{array}{l}\forall a.([a]p \leftrightarrow \\ ((a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p)))))\end{array}$$

Therefore we would have e.g. $[shoot]\neg alive \leftrightarrow (loaded \vee (\neg alive \wedge \neg\bot))$, from which it follows by classical principles that $(\neg hasGun \wedge alive \wedge [shoot]\neg alive) \rightarrow loaded$.

This means that such SSAs do not take into account inexecutability: this issue must be managed "by hand" by introducing $Poss(shoot)$ atoms in the right places when proving consequences of SSAs in their recent version.

Finally, we note that Reiter's presentation also contains precondition axioms of the form $Poss(\vec{a}) \leftrightarrow \varphi$. This is not needed here because we view $Poss(.)$ as a function returning a classical formula φ, which is directly integrated into our Successor State Axiom (cf. Section 5.2).

Reiter's Regression

Successor State Axioms are crucial when it comes to the reasoning aspect of the frame problem, to which we turn now.

Given a Reiter's style action theory $\mathcal{D}_{\mathsf{R}}$, what can be deduced from it? Suppose that Φ is a complex formula without quantification, action variables, and equality, such as, for example, $hasGun \rightarrow [load][shoot]\neg alive$. In order to decide whether $\mathcal{D}_{\mathsf{R}} \models \Phi$, Reiter proposes to rewrite Φ using the Successor State Axioms from the left to the right. This is what he calls *regression*, and it consists in syntactical substitutions whose iteration reduces a given formula with action symbols into another one with just propositional constants. The whole procedure is given in Algorithm 5.1.

At each regression step, we have to put formulas in normal form such that there are neither conjunctions nor disjunctions in the scope of modal operators (using the hypothesis that all actions are deterministic). Hence the innermost modal operators have just literals in their scope. For the above example, Φ gets $\neg hasGun \vee [load][shoot]\neg alive$.

Algorithm 5.1 Reiter's regression

input: a DPDL^+ formula Φ with no variables, $Poss(.)$, $causes^+(.)$, $causes^-(.)$, $Cond^+(.)$ and $Cond^-(.)$
output: a classical formula $regression(\Phi)$

```
while Φ is not classical do
  put Φ in normal form
  choose a subformula [a]ℓ
  if ℓ = p then
    replace [a]p by reg(a, p)
  else
    replace [a]¬p by reg(a, ¬p)
```

Notice that the action variable a of the Successor State Axiom is instantiated by the constant denoted by $\vec{a}$.

In our example, the regression of the subformula $[shoot]\neg alive$ is

$$\neg hasGun \vee (shoot = shoot \wedge loaded) \vee$$
$$(shoot = strangle \wedge \top) \vee (\neg alive \wedge \neg \bot)$$

This can be simplified to $\neg hasGun \vee loaded \vee \neg alive$. Hence the result of a one step regression of Φ is $\neg hasGun \vee [load](\neg hasGun \vee loaded \vee \neg alive)$.

Each rewriting step thus eliminates a modal operator, and iterated application results in a formula without modal operators. If we iterate regression in our example, we first put the formula

$$\neg hasGun \vee [load](\neg hasGun \vee loaded \vee \neg alive)$$

into normal form, obtaining

$$\neg hasGun \vee [load]\neg hasGun \vee [load]loaded \vee [load]\neg alive.$$

The regression of subformula $[load]\neg hasGun$ is equivalent to $\neg hasGun$, that of subformula $[load]loaded$ to $\top$, and that of $[load]\neg alive$ to $\neg alive$. We therefore obtain

$$\neg hasGun \vee \neg hasGun \vee \top \vee \neg alive,$$

which is valid in classical propositional logic. This means that the original formula $hasGun \rightarrow [load][shoot]\neg alive$ is entailed by $\mathcal{D}_{\mathsf{R}}$.

As regression is proved to be sound [100, Theorem 4.5.2], checking validity of the original formula amounts to checking satisfiability of the regressed one in the initial state of the world:

Theorem 5.2 ([100])
Let $\mathcal{D}_{\mathsf{R}}$ be a Reiter style domain description, and Φ be a formula without variables. Then $\mathcal{D}_{\mathsf{R}} \models \Phi \leftrightarrow regression(\Phi)$.

Corollary 5.1
$\mathcal{D}_{\mathsf{R}} \models \Phi$ if and only if $\models_{\overline{\mathsf{CPL}}} regression(\Phi)$.

In the rest of this chapter, we explore whether regression can be performed in a simpler framework, in particular without quantifying over actions.

5.4 Solving the Frame Problem without Quantification

The venue of Reiter's Situation Calculus-based solution has raised the natural question of at what extent it could be possible to do the same in dynamic logic. Given the expressivity limitations of the latter w.r.t. first-order logic (originally it did not allow for quantification over actions), many researchers [119, 120] have turned to other ways of facing the problems in the area. There has been others [23], however, who have tried on the first steps in that direction.

We here give DPDL$^+$ up and consider just deterministic PDL and possible extensions of it in order to encode Reiter's solution to the frame problem.

De Giacomo and Lenzerini's Encoding into PDL

De Giacomo and Lenzerini [23] have expressed Reiter's solution in a slightly modified version of PDL that avoids quantification over actions. For the sake of presentation, here we simplify their account a bit. Basically, their approach can be said to have the following ingredients (α denotes a complex action, i.e., an action built up on atomic actions and PDL classical action composition operators):

- nondeterministic choice $\alpha \cup \alpha'$;
- converse α^-;
- a particular nondeterministic atomic action **any**, thought of as the nondeterministic composition of all atomic actions of $\mathfrak{Act}$: $\mathbf{any} = a_1 \cup a_2 \cup \ldots \cup a_n$; and
- complement $\neg\alpha$ w.r.t. **any**, where $\alpha = a_1 \cup \ldots \cup a_m$, for some $a_1, \ldots, a_m \in \mathfrak{Act}$.

Moreover, it is supposed that the past is deterministic, as expressed by the logical axiom $\neg[\mathbf{any}^-]\neg\Phi \rightarrow [\mathbf{any}^-]\Phi$.

Considering our running example, its formalization in De Giacomo and Lenzerini's framework would be:

$$[\mathbf{any}](\neg \mathit{alive} \rightarrow \langle \mathbf{any}^- \rangle \neg \mathit{alive} \vee \langle \mathit{shoot}^- \rangle \mathit{loaded} \vee \langle \mathit{strangle}^- \rangle \top)$$

$$[\mathbf{any}](\mathit{alive} \rightarrow \langle \mathbf{any}^- \rangle \mathit{alive})$$

Just as for PDL, reasoning in De Giacomo and Lenzerini's logical framework is EXPTIME-complete [23]. While their encoding certainly preserves the spirit of Reiter's Successor State Axioms, they did not give the counterpart of Reiter's regression,

and hence did not investigate whether reasoning for syntactically restricted theories is "cheaper" than EXPTIME. In what follows, we show how this can be simulated without quantification in the dependence-based framework we introduced in Chapter 4.

Regression in PDL plus Dependence

We start by observing that stating $a \rightsquigarrow p$ in the dependence-based framework is just another way of writing down that $a \in \mathit{causes}^+(p)$, and $a \rightsquigarrow \neg p$ that $a \in \mathit{causes}^-(p)$.

Suppose all the ingredients $\mathit{Poss}(.)$, $\mathit{causes}^+(.)$, $\mathit{causes}^-(.)$, $\mathit{Cond}^+(.)$, $\mathit{Cond}^-(.)$ are given, and let us make the completeness assumptions as introduced in Section 5.2. We construct a dependence relation and a set of global axioms $\mathcal{T}$ as follows:

- for each $p \in \mathfrak{Prop}$: for every $a \in \mathit{causes}^+(p)$, we put $a \rightsquigarrow p$; and for every $a' \in \mathit{causes}^-(p)$, we put $a' \rightsquigarrow \neg p$;
- for every $a \in \mathfrak{Act}$, add the executability axiom $\mathit{Poss}(a) \leftrightarrow \neg[a]\bot$ to $\mathcal{T}$
- for every $p \in \mathfrak{Prop}$ and every $a \in \mathit{causes}^+(p)$, add two effect axioms to $\mathcal{T}$:

$$\mathit{Cond}^+(a, p) \rightarrow [a]p \tag{5.5}$$

$$(\neg \mathit{Cond}^+(a, p) \wedge \neg p) \rightarrow [a]\neg p \tag{5.6}$$

- for every $p \in \mathfrak{Prop}$ and every $a' \in \mathit{causes}^-(p)$, add two effect axioms to $\mathcal{T}$:

$$\mathit{Cond}^-(a', p) \rightarrow [a']\neg p \tag{5.7}$$

$$(\neg \mathit{Cond}^-(a', p) \wedge p) \rightarrow [a']p \tag{5.8}$$

Note that these axioms do not resemble Successor State Axioms. They nevertheless validate the same regression principle as in Reiter's framework, as it will be shown in the sequel.

A point that bears noting is that our representation indeed counts as a solution to the frame problem: the sets $\rightsquigarrow$ and $\mathcal{T}$ are both "small" (in the sense that we can expect they are much smaller than $\mathit{card}(\mathfrak{Prop}) \times \mathit{card}(\mathfrak{Act})$), and contain no frame axioms.

Now we turn to an important result:

Theorem 5.3

Let the underlying logic be deterministic PDL, $\rightsquigarrow$ *be a dependence relation obtained from sets* $causes^+(.)$ *and* $causes^-(.)$*, and let* $\mathcal{T}$ *be the set of global axioms (5.4)–(5.8). Then*

(1) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee p$*, if* $a \not\rightsquigarrow p$ *and* $a \not\rightsquigarrow \neg p$*;*

(2) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee (p \wedge \neg Cond^-(a,p))$*, if* $a \not\rightsquigarrow p$ *and* $a \rightsquigarrow \neg p$*;*

(3) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee Cond^+(a,p) \vee p$*, if* $a \rightsquigarrow p$ *and* $a \not\rightsquigarrow \neg p$*; and*

(4) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee Cond^+(a,p) \vee (p \wedge \neg Cond^-(a,p))$*, if* $a \rightsquigarrow p$ *and* $a \rightsquigarrow \neg p$*.*

Proof:

See Appendix B. ■

Based on this result, with Algorithm 5.2 we give a regression method for deterministic PDL with a dependence relation. (Let us consider $Cond(a,\ell) = Cond^+(a,p)$, if $\ell = p$, and $Cond(a,\ell) = Cond^-(a,p)$, if $\ell = \neg p$.)

Algorithm 5.2 Regression with dependence

input: a PDL formula Φ, $Poss(.)$, $causes^+(.)$, $causes^-(.)$, $Cond^+(.)$ and $Cond^-(.)$
output: a classical formula $regression(\Phi)$

```
while Φ is not classical do
  put Φ in normal form
  choose some subformula [a]ℓ
  case a ↛ ℓ and a ↛ ¬ℓ
    replace [a]ℓ by ¬Poss(a) ∨ ℓ
  case a ↛ ℓ and a ⇝ ¬ℓ
    replace [a]ℓ by ¬Poss(a) ∨ (ℓ ∧ ¬Cond(a, ¬ℓ))
  case a ⇝ ℓ and a ↛ ¬ℓ
    replace [a]ℓ by ¬Poss(a) ∨ Cond(a, ℓ) ∨ ℓ
  case a ⇝ ℓ and a ⇝ ¬ℓ
    replace [a]ℓ by ¬Poss(a) ∨ Cond(a, ℓ) ∨ (ℓ ∧ ¬Cond(a, ¬ℓ))
```

Suppose Φ is a complex formula without quantification and equality, such as $hasGun \rightarrow [load][shoot]\neg alive$. Then, running Algorithm 5.2 on Φ, the regression of $[shoot]\neg alive$ is $\neg hasGun \vee loaded \vee \neg alive$. Hence the result of this regression step is $hasGun \rightarrow [load](\neg hasGun \vee loaded \vee \neg alive)$. Putting this into normal form using axiom (5.3), we obtain the formula $hasGun \rightarrow ([load]\neg hasGun \vee [load]loaded \vee [load]\neg alive)$. The regression of $[load]\neg hasGun$ is $\neg hasGun$, that of $[load]loaded$ is $\top$, and that of $[load]\neg alive$ is $\neg alive$. We therefore obtain $hasGun \rightarrow (\neg hasGun \vee \top \vee \neg alive)$, which is valid in classical propositional logic.

Theorem 5.4 (Decidability, soundness and completeness)
Let $\mathcal{T}$ and $\rightsquigarrow$ be obtained from $Poss(.)$, $causes^{+}(.)$, $causes^{-}(.)$, $Cond^{+}(.)$ and $Cond^{-}(.)$, and let Φ be a complex formula. Then, Algorithm 5.2 terminates returning a classical formula φ and $\mathcal{T} \models_{\rightsquigarrow} \Phi \leftrightarrow \varphi$.

Proof:
Let Φ be an input formula. Termination is straightforward, as each step of the algorithm eliminates exactly one modal operator. Soundness and completeness are also immediate: after putting formula Φ in normal form, it will be made of conjunctions/disjunctions of modal subformulas. In this case, the equivalence between Φ and φ follows from the ones given in Theorem 5.3 together with the rule of substitution of equivalences (which is valid in PDL). ■

For our running example, $hasGun \rightarrow [load][shoot]\neg alive$ is a consequence of the theory $\mathcal{T}$ with the dependence relation $\rightsquigarrow$ because its regression is classically valid.

Hence, modulo equality, we obtain the same result as for Reiter's regression in our example. This generalizes: a close look at both algorithms shows that if both our $\mathcal{T}$ with $\rightsquigarrow$ and Reiter's domain description are obtained from the same $Poss(.)$, $causes^{+}(.)$, $causes^{-}(.)$, $Cond^{+}(.)$, $Cond^{-}(.)$, then the results are logically equivalent.

It follows thus that whenever $Poss(.)$, $causes^{+}(.)$, $causes^{-}(.)$, $Cond^{+}(.)$, $Cond^{-}(.)$ are given, and the completeness assumptions can be made, then Reiter's formulation in terms of Successor State Axioms and ours in terms of effect axioms and dependence do the same job in their respective logical basis:

Corollary 5.2
Let the sets $Poss(.)$, $causes^{+}(.)$, $causes^{-}(.)$, $Cond^{+}(.)$, $Cond^{-}(.)$ be given. Let $\mathcal{D}_{\mathsf{R}} = \langle \mathcal{L}_{\mathsf{DPDL}^{+}}, \models_{\mathsf{R}}, \mathcal{T}_{\mathsf{R}} \rangle$ be a Reiter theory obtained from them as described in Section 5.3, and let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$ be obtained from them as described above. Let Φ be a complex formula without quantification and equality. Then $\mathcal{D}_{\mathsf{R}} \models \Phi$ if and only if $\mathcal{D} \models \Phi$.

Proof:
Straightforward. ■

5.5 What about the Ramification Problem?

Reiter's solution supposes that domain descriptions only contain executability and effect laws. Thus it does not allow for static laws such as $walking \rightarrow alive$. Such laws

augment the effects of the *shoot* action: shooting not only has the (direct) effect $\neg alive$, but also the (indirect) effect $\neg walking$ (Figure 5.2).

Figure 5.2: Indirect effect of shooting: the turkey stops walking.

Because an action can have too many indirect effects, stating all of them in the form of effect laws may be unfeasible and make the domain description unmanageable. The problem of being able to derive all indirect effects of an action without explicitly stating them as axioms is known as the *ramification problem* [32]. Basically, this one states that we should not relate actions with their indirect effects (in the base logic).

In the example above, instead of stating the law $loaded \rightarrow [shoot]\neg walking$ in the theory, we should rather be able to conclude that ramification just from the base effect laws for *shoot* and the static law $walking \rightarrow alive$. Nevertheless, as extensively addressed in the literature, static laws alone are not enough to express a notion of *causation* and thus cannot cope to avoid the derivation of indirect effects not properly caused by the action under consideration. For example, from $[tease]walking$ and $walking \rightarrow alive$ in our scenario, we conclude that *alive* is always true after execution of action *tease*, which intuitively may not be the case if *alive* initially does not hold: teasing a dead turkey does not resurrect it.

This means that the only indirect effects that follow from an action theory should be those that are really relevant.

In the recent literature on reasoning about actions, the concept of *causality* has been studied as a means of overcoming the inadequacy of static laws in tackling the ramification problem. In this sense, many types of causal notions have been proposed so that causality is then considered in different ways: *strong* [78, 83, 119, 64] or *weak* [112, 14] causality (if we always *force* or only *permit* something to be caused); as a *predicate* [78], a *relation* [112, 14] or a *modality* [84, 45, 43, 35]; and *primitive* (built in the logic) [78, 43, 119] or *derived* (with the aid of some meta-logical information) [112, 14].

The dependence-based solution to the frame problem we have chosen is an example of a meta-logical causal notion expressed as a relation. With it we avoid the problem of stating indirect effects in the base logic without unintuitive results. In our example, by stating the dependence $\mathit{shoot} \rightsquigarrow \neg \mathit{walking}$ we get *shoot*'s indirect effect, and because $\mathit{tease} \not\rightsquigarrow \mathit{alive}$, we do not get *alive* as indirect effect of *tease*.

Basically, all the extant approaches in the literature perform well in describing dynamic domains where ramifications have to be dealt with. Nevertheless, things get more complicated when actions with both indeterminate and indirect effects are involved. This is what we are going to address in the following chapter.

Chapter 6

where we show what others cannot do

Causality and Indeterminate Indirect Effects

In the sharp formulation of the law of causality – 'if we know the present exactly, we can calculate the future' – it is not the conclusion that is wrong, but the premise.

— Heisenberg

In this chapter, we investigate the behavior of the main existing causal approaches to reasoning about actions that are called fluent-indexed frameworks. In particular, we analyze how they perform in dealing with domains that have actions with both nondeterministic and indirect effects. For this, we present an example of such a scenario, give a requirement concerning the interpretation of indeterminate indirect effects and study it through the chapter.

6.1 The Mailboxes Scenario

Many approaches consider that it is a change in some property that produces (causes) change of some other property. We call them *fluent-indexed approaches*, for they relate pairs of literals or formulas.

We argue here that fluent-indexed approaches are not enough for dealing with the ramification problem in domains involving actions with both nondeterministic and indirect effects. We do this by showing an example of this class of action domain that will lead us through a systematic analysis of such approaches.

We present here the Mailboxes Scenario, which was originally defined in [15].

In essence, it combines Reiter's famous "dropping a coin on a chessboard" example with Sandewall's argument against causality-based solutions to the ramification problem [101].

In such a scenario, we reason about the status of a particular e-mail message and two mailboxes (Figure 6.1).

Figure 6.1: The Mailboxes Scenario.

The domain is as follows: suppose *mbox*1 means "the message is in mailbox 1", and *mbox*2 "the message is in mailbox 2". We represent the fact that the e-mail is saved in *mbox*1 or in *mbox*2 or in both by the literal *saved*. Hence the static law for this example is

$$saved \leftrightarrow (mbox1 \vee mbox2)$$

in formalisms that are not situation-indexed, and

$$Holds(saved, s) \leftrightarrow (Holds(mbox1, s) \vee Holds(mbox2, s))$$

in situation-indexed formalisms such as the Situation Calculus. (As usual, we assume that all free variables denoting situations are universally quantified.)

Consider the actions *save*1 and *save*2, whose direct effects are to save an e-mail message in *mbox*1 and in *mbox*2, respectively. Suppose we also have a nondeterministic *save* action, whose direct effect is *saved*, i.e., saving the e-mail in one of the two mailboxes or in both. Hence *save* has the indirect effect $mbox1 \vee mbox2$. This is also an indeterminate effect. Note that, in particular, after executing *save*, it is also possible

to have $mbox1 \wedge mbox2$. This is just as in Reiter's "dropping a coin on a chessboard" example, where *drop* has the possible effect $black \wedge white$.[1]

In order to correctly reason about a nondeterministic action, we have to be able of properly treating its set of indeterminate effects. This means that we should not systematically interpret effects described with the inclusive disjunction '$\vee$' as the exclusive one '$\oplus$'. For example, in the Mailboxes Scenario, the effect of *save* should not be equivalent to $mbox1 \oplus mbox2$. The motivation for such a requirement has been originally suggested by Reiter.

As we will see along this chapter, the Mailboxes Scenario is problematic for all the existing approaches allowing for the representation of actions with both indirect and indeterminate effects. In what follows, we discuss the approaches of Lin [78, 79], McCain and Turner [83, 84], Thielscher [112, 113] and Zhang and Foo [119]. Indeed, it can be shown that, in all these frameworks, either we have to state a frame axiom, or to relate an action (in the base logic) with some of its ramifications, or, in order not to violate our requirement about the interpretation of disjunctions, the action $save1$ has the indirect indeterminate effect of changing $mbox2$, which is clearly counterintuitive.

6.2 Minimization of Causality

We here examine the behavior of Lin's causal approach [78, 79] in formalizing the Mailboxes Scenario.

Roughly speaking, Lin proposes to add a new predicate $Caused(.)$ to the Situation Calculus. $Caused(p, v, s)$ reads as "atom p is caused to have truth value v in situation s". Such a predicate is used to describe the appropriate causal relationships between fluents. In order to solve the frame problem, instances of $Caused(.)$ shall be minimized via circumscription [86, 87, 76].

In addition, the following axioms are assumed:

$$Caused(p, true, s) \rightarrow Holds(p, s) \tag{6.1}$$

$$Caused(p, false, s) \rightarrow \neg Holds(p, s) \tag{6.2}$$

[1] It is possible as well to rephrase our example in terms of Reiter's: we can regard action *save* as *drop*, which means putting a pin on a white, a black, or both squares (the pin lying on the region between two squares). $save1$ (resp. $save2$) can be seen as analogous to $drop1$ (resp. $drop2$), which means putting the pin in a black (resp. white) square.

which state that something that is caused in a situation s must hold in such a situation, as well as something that is caused to cease is no longer valid in that situation.

In what follows, we describe the Mailboxes Scenario using this formalism. Following the definitions in the original work, the effect axioms for this scenario are:

$$Poss(save1, s) \rightarrow Caused(mbox1, true, do(save1, s)) \tag{6.3}$$

$$Poss(save2, s) \rightarrow Caused(mbox2, true, do(save2, s)) \tag{6.4}$$

$$Poss(save, s) \rightarrow Caused(saved, true, do(save, s)) \tag{6.5}$$

Then, according to Lin's method, we have to supplement the static law $saved \leftrightarrow (mbox1 \vee mbox2)$ in the following way: as $save1$ (resp. $save2$) has effect $mbox1$ (resp. $mbox2$) and $mbox1$ (resp. $mbox2$) being true causes the truth of $saved$, then we must causally relate $mbox1$ (resp. $mbox2$) and $saved$. This is done stating the formulas:

$$Caused(mbox1, true, s) \rightarrow Caused(saved, true, s) \tag{6.6}$$

$$Caused(mbox2, true, s) \rightarrow Caused(saved, true, s) \tag{6.7}$$

Thus, the way domain constraints and effect axioms are stated defines a fluent-indexed strong causal notion: an atom being causally related with another, whenever it becomes true, the other is forced to become true.

The other way round, as an execution of *save* has the direct effect *saved* and a change in *saved* means a change in $mbox1$ and/or in $mbox2$, we are obliged to causally relate *saved* with both $mbox1$ and $mbox2$. This is done stating the formula:

$$Caused(saved, true, s) \rightarrow Caused(mbox1, true, s) \vee Caused(mbox2, true, s) \tag{6.8}$$

Stating just these laws, according to the circumscription-based minimization process defined in [79], we would get an exclusive interpretation of the disjunction in (6.8), i.e., *save* would have the indirect effect $mbox1 \oplus mbox2$. So, in order to capture the possibility of *save* saving the e-mail in both mailboxes, in Lin's approach we have also to state the constraints:[2]

$$Caused(saved, true, s) \rightarrow Caused(mbox1, true, s) \vee Caused(mbox1, false, s) \tag{6.9}$$

$$Caused(saved, true, s) \rightarrow Caused(mbox2, true, s) \vee Caused(mbox2, false, s) \tag{6.10}$$

[2]It is worth noting that both consequents of (6.9) and (6.10) are not tautologies (cf. [78]).

Thus, we have the following:

Proposition 6.1
Formulas (6.3)–(6.10) entail

$$Poss(save1, s) \rightarrow \quad Caused(mbox2, true, do(save1, s)) \vee \\ Caused(mbox2, false, do(save1, s))$$

Proof:
Suppose that $Poss(save1, s)$ is the case. Then, from Formula (6.3) we obtain $Caused(mbox1, true, s')$, where s' stands for $do(save1, s)$. From this and Formula (6.6), we get $Caused(saved, true, s')$. Thus, constraint (6.9) gives us $Caused(mbox1, true, s') \vee Caused(mbox1, false, s')$. Nevertheless, even with the minimization policy defined in [79], it is still possible to derive another extension: from $Caused(saved, true, s')$ and constraint (6.10) we conclude $Caused(mbox2, true, s') \vee Caused(mbox2, false, s')$. ■

So, we get that an execution of *save*1 can produce the indirect effect of changing *mbox*2. But we do not want such an indirect effect, for *save*1 would be nondeterministic. A possible solution for this could be to state

$$(Poss(save1, s) \wedge \neg Holds(mbox2, s)) \rightarrow Caused(mbox2, false, do(save1, s))$$

from which we derive

$$(Poss(save1, s) \wedge \neg Holds(mbox2, s)) \rightarrow \neg Holds(mbox2, do(save1, s))$$

but this is a frame axiom.

Another tentative of tackling the problem is stating

$$Poss(save1, s) \rightarrow Caused(mbox2, false, do(save1, s))$$

but, this is unintuitive, for in a situation where we already had *saved*, with the e-mail in *mbox*2, saving again with *save*1 would make a change in *mbox*2.

6.3 Causal Laws Approach

In this section, we formalize the Mailboxes Scenario using the base formalism proposed by McCain and Turner [83]. Their approach considers that background knowl-

edge about causation should be given in the form of *causal laws*, which are stated as sentences in a modal, conditional logic with the aid of a causal modal operator $\Rightarrow$.

A causal law of the form $\varphi \Rightarrow \psi$, where φ and ψ are classical formulas, is read as "φ causes ψ", or "the truth of φ determines the truth of ψ". In our terms, this is thus a fluent-indexed causal approach.

Let **Laws** be the set of all causal laws concerning a given domain. A set of formulas $\mathcal{T}$ is *closed* under **Laws** if and only if whenever $\varphi \Rightarrow \psi$ is in **Laws** and $\varphi \in \mathcal{T}$, then $\psi \in \mathcal{T}$. $\mathcal{T} \vdash_{\mathbf{Laws}} \varphi$ means that formula φ belongs to the smallest set of formulas containing $\mathcal{T}$ that is closed w.r.t. propositional logic and also closed under **Laws**.

In the formalization that follows, a set of literals **Facts** denotes a knowledge base (alias state), and **Eff** a set of direct effects.

With the causal laws approach, the representation of the Mailboxes Scenario is as follows:

$$\mathbf{Laws} = \left\{ \begin{array}{c} saved \Rightarrow (mbox1 \vee mbox2), \\ (mbox1 \vee mbox2) \Rightarrow saved \end{array} \right\}$$

The causal law $saved \Rightarrow (mbox1 \vee mbox2)$ is needed because the truth of fluent *saved* causes the truth of formula $mbox1 \vee mbox2$. Analogously, $(mbox1 \vee mbox2) \Rightarrow saved$ is necessary because $mbox1 \vee mbox2$ being true causes *saved* also to be true. (Instead of $(mbox1 \vee mbox2) \Rightarrow saved$ one could have as well the causal laws $mbox1 \Rightarrow saved$ and $mbox2 \Rightarrow saved$, whose justifications are straightforward. On the other hand, we could not replace $saved \Rightarrow (mbox1 \vee mbox2)$ by $saved \Rightarrow mbox1$ and $saved \Rightarrow mbox2$, for in this case *save* would always cause $mbox1 \wedge mbox2$.)

Completing the domain description, we have a set of initial observations:

$$\mathbf{Facts}_0 = \{\neg mbox1, \neg mbox2, \neg saved\}$$

and we suppose that *saved* has been produced as a direct effect:

$$\mathbf{Eff} = \{saved\}$$

From this representation and according to McCain and Turner's approach defined in [83], after *save* action we get an exclusive interpretation of the disjunction $mbox1 \vee mbox2$. This is shown in the following proposition:

Proposition 6.2

Let $\mathbf{Facts}_0 = \{\neg mbox1, \neg mbox2, \neg saved\}$ *and* $\mathbf{Eff} = \{saved\}$*. Then the only possible successor states are:*

$$\left\{ \begin{array}{l} \{mbox1, \neg mbox2, saved\}, \\ \{\neg mbox1, mbox2, saved\} \end{array} \right\}$$

Proof:

Following the definitions in [83], for any knowledge base **Facts**, any direct effects **Eff**, and any set **Laws** of causal laws, the set of possible next states after performing an action is the set of interpretations $\mathbf{Facts}'$ such that:

$$\mathbf{Facts}' = \{\ell \ : \ \ell \in \mathfrak{Lit}, (\mathbf{Facts} \cap \mathbf{Facts}') \cup \mathbf{Eff} \mid\!\sim_{\mathbf{Laws}} \ell\}$$

where $\mid\!\sim_{\mathbf{Laws}}$ is derivability w.r.t. the causal laws defined in **Laws**.

For the possible next state $\mathbf{Facts}_1 = \{mbox1, \neg mbox2, saved\}$, we have $\mathbf{Facts}_0 \cap \mathbf{Facts}_1 = \{\neg mbox2\}$ and $\{\neg mbox2\} \cup \{saved\} \mid\!\sim_{\mathbf{Laws}} mbox1$, and this is a possible next state. For the state $\mathbf{Facts}_2 = \{\neg mbox1, mbox2, saved\}$, we have $\mathbf{Facts}_0 \cap \mathbf{Facts}_2 = \{\neg mbox1\}$ and $\{\neg mbox1\} \cup \{saved\} \mid\!\sim_{\mathbf{Laws}} mbox2$, and this is a possible next state, too. The interpretation $\mathbf{Facts}_3 = \{\neg mbox1, \neg mbox2, saved\}$ is not a possible next state as clearly $\mathbf{Facts}_3$ is not closed under **Laws**. Now, considering the state $\mathbf{Facts}_4 = \{mbox1, mbox2, saved\}$, we have $\mathbf{Facts}_0 \cap \mathbf{Facts}_4 = \emptyset$ and neither $\emptyset \cup \{saved\} \not\mid\!\sim_{\mathbf{Laws}} mbox1$ nor $\emptyset \cup \{saved\} \not\mid\!\sim_{\mathbf{Laws}} mbox2$, so $\mathbf{Facts}_4$ is not closed under **Laws**. Thus, the only possible states after performing the *save* action are $\mathbf{Facts}_1$ and $\mathbf{Facts}_2$, and from this the result follows.[3] ■

In order to avoid exclusive interpretation of disjunctions, we have to relax inertia by increasing **Laws** with the following causal laws

$$(saved \wedge mbox1) \Rightarrow mbox1$$
$$(saved \wedge mbox2) \Rightarrow mbox2$$

However, with this apparent solution we get that an execution of $save1$ could make a change in $mbox2$: the interpretation $\{mbox1, mbox2, saved\}$ is a possible next state of $\mathbf{Facts}_0$ w.r.t. $\mathbf{Eff} = \{mbox1\}$.

In [84] an improved version of the causal laws approach is given. Basically, the difference is that actions are made explicit and each action, fluent and formula has an associated time point. For example, $save1_2$ means that the action of saving the e-mail

[3] The reader is invited to verify that with the causal laws $mbox1 \Rightarrow saved$ and $mbox2 \Rightarrow saved$ instead of $(mbox1 \vee mbox2) \Rightarrow saved$ one obtains the same result.

in mailbox 1 is executed at time point 2, and having $mbox1_3$ means that at time point 3, the e-mail is saved in mailbox 1 (independently of the action that has been executed to achieve that).

Besides considering time, the following *standard schemas* are also assumed (remembering, a stands for action names, p for atom (fluent) names, and φ for a formula):

$$a_t \Rightarrow a_t \tag{6.11}$$

$$\neg a_t \Rightarrow \neg a_t \tag{6.12}$$

$$p_0 \Rightarrow p_0 \tag{6.13}$$

$$\neg p_0 \Rightarrow \neg p_0 \tag{6.14}$$

$$\varphi_t \wedge \varphi_{t+1} \Rightarrow \varphi_{t+1} \tag{6.15}$$

Schema (6.11) (resp. (6.12)) states that the occurrence (resp. non-occurrence) of action a at time t is caused whenever a occurs (resp. does not occur) at t. The Schemas (6.13) and (6.14) establish that the initial observations are caused from the beginning. Schema (6.15) formalizes the common sense law of inertia, representing the fact that whenever a set of fluents holds at two successive time points, their truth at the second time point is taken to be caused simply by virtue of its persistence.

Using this variant of the causal laws approach, we formalize the Mailboxes Scenario in the following way (**Laws**, $\mathbf{Facts}_0$ and **Eff** are as above, except that they are time-indexed):

$$\mathbf{Laws} = \left\{ \begin{array}{c} save1_t \wedge \neg mbox1_t \Rightarrow mbox1_{t+1}, \\ save2_t \wedge \neg mbox2_t \Rightarrow mbox2_{t+1}, \\ save_t \Rightarrow saved_{t+1}, \\ saved_t \Rightarrow (mbox1_t \vee mbox2_t), \\ (mbox1_t \vee mbox2_t) \Rightarrow saved_t \end{array} \right\}$$

$$\mathbf{Facts}_0 = \{\neg mbox1_0, \neg mbox2_0, \neg saved_0\}$$

Again, with such a representation, our requirement about the interpretation of the disjunction is violated: we get an exclusive interpretation of the nondeterminism of the *save* action. As before, if we relax inertia by means of some extra causal laws, we will also get that *save*1 may cause a change in *mbox*2.

6.4 Postprocessing Approach

In this section, we examine the postprocessing generation of ramifications proposed by Thielscher [112, 113]. The basic idea of this approach consists in admitting states not satisfying the domain constraints, which are seen as "intermediate states". "Stable" states are obtained after successive applications of the so called *causal relations*. A causal relation ℓ_1 causes ℓ_2 if φ, where $\ell_1, \ell_2 \in \mathfrak{Lit}$ and $\varphi \in \mathfrak{Fml}$, is the way a fluent indexed causal notion is defined in this approach.

In what follows, an *action law* is a triple $\langle \mathsf{C}, a, \mathsf{E} \rangle$, where a is an action, and C and E are sets of literals containing, respectively, the action preconditions and effects, and such that $atm(\mathsf{C}) = atm(\mathsf{E})$ (C and E have the same atoms). An *influence relation* is a relation between atoms that is used to automatically generate the causal relations. Saying that a pair (p_1, p_2), where $p_1, p_2 \in \mathfrak{Prop}$, is in the influence relation means that a change in the truth value of p_1 *may* cause a change in the truth value of p_2.

A state of the world (not necessarily satisfying the domain constraints) is a pair of sets of literals $(\mathbf{Facts}, \mathbf{Eff})$, where **Facts** denotes a knowledge base and **Eff** a set of direct effects. An action law $\langle \mathsf{C}, a, \mathsf{E} \rangle$ is *applicable* to a state $(\mathbf{Facts}, \mathbf{Eff})$ if and only if $\mathsf{C} \subseteq \mathbf{Facts}$. Performing an action a in a state of affairs **Facts** corresponds to applying its associated action law $\langle \mathsf{C}, a, \mathsf{E} \rangle$ to the pair $(\mathbf{Facts}, \mathbf{Eff})$, giving us a new pair $(\mathbf{Facts}', \mathbf{Eff}')$, where $\mathbf{Facts}' = (\mathbf{Facts} \setminus \mathsf{C}) \cup \mathsf{E}$ and $\mathbf{Eff}' = \mathbf{Eff} \cup \mathsf{E}$.

A causal relation ℓ_1 causes ℓ_2 if φ is *applicable* to a state $(\mathbf{Facts}, \mathbf{Eff})$ if and only if $\mathbf{Facts} \models_{\mathsf{CPL}} \varphi \wedge \neg\ell_2 \wedge \ell_1$ and $\ell_1 \in \mathbf{Eff}$. The state resulting from applying such a causal relation is $(\mathbf{Facts}', \mathbf{Eff}')$, where $\mathbf{Facts}' = (\mathbf{Facts} \setminus \{\neg\ell_2\}) \cup \{\ell_2\}$ and $\mathbf{Eff}' = (\mathbf{Eff} \setminus \{\neg\ell_2\}) \cup \{\ell_2\}$.

For the Mailboxes Scenario, we define the following action laws:

$$\langle \{\neg mbox1\}, save1, \{mbox1\} \rangle \tag{6.16}$$

$$\langle \{\neg mbox2\}, save2, \{mbox2\} \rangle \tag{6.17}$$

$$\langle \{\neg saved\}, save, \{saved\} \rangle \tag{6.18}$$

Action law (6.16) expresses that "in a state where *mbox*1 is false, after executing *save*1, *mbox*1 will be true". For action laws (6.17) and (6.18), the reading is analogous.

The set of static laws is the singleton $\{saved \leftrightarrow (mbox1 \vee mbox2)\}$.

According to Thielscher's approach, as for this example a change in *mbox*1 (resp. *mbox*2) may cause a change in *saved* and vice-versa, we have to define the influence

relation for this scenario as follows:

$$\left\{ \begin{array}{l} (mbox1, saved),\ (mbox2, saved), \\ (saved, mbox1),\ (saved, mbox2) \end{array} \right\}$$

From this influence information and Algorithm 1 given in [112], we obtain the following set of causal relations:

$$\left\{ \begin{array}{c} saved \text{ causes } mbox1 \text{ if } \neg mbox2, \\ saved \text{ causes } mbox2 \text{ if } \neg mbox1, \\ \neg mbox1 \text{ causes } \neg saved \text{ if } \neg mbox2, \\ \neg mbox2 \text{ causes } \neg saved \text{ if } \neg mbox1, \\ mbox1 \text{ causes } saved \text{ if } \top, \\ \neg saved \text{ causes } \neg mbox1 \text{ if } \top, \\ mbox2 \text{ causes } saved \text{ if } \top, \\ \neg saved \text{ causes } \neg mbox2 \text{ if } \top \end{array} \right\}$$

Thus, with this domain description, we get the following:

Proposition 6.3
Let $(\{\neg mbox1, \neg mbox2, \neg saved\}, \emptyset)$ *be an initial state. Then the only possible successor states after executing save action are* $(\{mbox1, \neg mbox2, saved\}, \{saved, mbox1\})$ *and* $(\{\neg mbox1, mbox2, saved\}, \{saved, mbox2\})$.

Proof:
Let $(\{\neg mbox1, \neg mbox2, \neg saved\}, \emptyset)$ be the initial state. Then, applying the action law (6.18) to it, we get the resulting (intermediate) state

$$(\{\neg mbox1, \neg mbox2, saved\}, \{saved\}) \tag{6.19}$$

As (6.19) is inconsistent w.r.t. the static law $saved \leftrightarrow (mbox1 \vee mbox2)$, we apply the causal relation $saved$ causes $mbox1$ if $\neg mbox2$ to (6.19) and obtain

$$(\{mbox1, \neg mbox2, saved\}, \{saved, mbox1\})$$

which is a successor state [112]. In this state, no other causal relation can be applied.

Looking at (6.19) again, we apply the causal relation $saved$ causes $mbox2$ if $\neg mbox1$, and obtain

$$(\{\neg mbox1, mbox2, saved\}, \{saved, mbox2\})$$

which is a successor state, too. Again, in such a state, no other causal relation is applicable. The same observation holds now for (6.19).

Therefore, there are only two successor states. ■

Then, we get that with Thielscher's approach, action *save* gives an exclusive interpretation of the conjunction in its indirect effects.

6.5 Modal Causality

We now formalize the Mailboxes Scenario using the base logic EPDL, proposed by Foo and Zhang [119, 35]. Essentially, such a logic is an extension of PDL that allows for modalities of the form $[\varphi]$, with $\varphi \in \mathfrak{Fml}$, for specifying the indirect effects of actions. Given $\varphi, \psi \in \mathfrak{Fml}$, the *causal statement* $[\varphi]\psi$ means that formula ψ is caused whenever φ is the case. The semantical counterpart of such an extension is that models are of the form $\langle W, R\rangle$, where W is as defined in Chapter 2, and $R : \mathfrak{Act} \cup \mathfrak{Fml} \longrightarrow 2^{W\times W}$ is a function mapping action constants a to accessibility relations $R_a \subseteq W \times W$, and classical formulas φ to accessibility relations $R_\varphi \subseteq W \times W$. Moreover, every EPDL-model $\mathscr{M} = \langle W, R\rangle$ must satisfy that for all $w \in W$ and every $\varphi \in \mathfrak{Fml}$, if $\models^{\mathscr{M}}_{w} \varphi$, then $wR_\varphi w$.

Therefore in EPDL we are able to write formulas like $[mbox1]saved$, which states that in all possible worlds in which $mbox1$ is true, *saved* is caused to be true. The complete domain description for the Mailboxes Scenario in EPDL is given bellow:

$$\mathcal{T} = \left\{ \begin{array}{c} [saved](mbox1 \vee mbox2), [mbox1 \vee mbox2]saved, \\ \langle save\rangle\top, \langle save1\rangle\top, \langle save2\rangle\top, \\ [save]saved, [save1]mbox1, [save2]mbox2 \end{array} \right\}$$

In Foo and Zhang's approach, static laws are implicitly derived from the causal statements. Then, for the theory above, we have $\mathcal{T} \models_{\mathsf{EPDL}} saved \leftrightarrow (mbox1 \vee mbox2)$ without explicitly stating it.

Proposition 6.4
$\mathcal{T} \models_{\mathsf{EPDL}} (\neg mbox1 \wedge \neg mbox2) \rightarrow [save1](mbox1 \vee mbox2)$.

Proof:

1. $\neg mbox1 \rightarrow [save1]mbox1$, from global axioms $\mathcal{T}$ and classical logic
2. $\neg mbox2 \rightarrow [save1]mbox1$, from global axioms $\mathcal{T}$ and classical logic
3. $[mbox1 \vee mbox2]saved$, from global axioms $\mathcal{T}$

4. $(mbox1 \vee mbox2) \rightarrow saved$, from 3. and EPDL
5. $\neg mbox2 \rightarrow [save1]saved$, from 2., 4. and classical logic
6. $[saved](mbox1 \vee mbox2)$, from global axioms $\mathcal{T}$
7. $saved \rightarrow (mbox1 \vee mbox2)$, from 6. and EPDL
8. $\neg mbox2 \rightarrow [save1](mbox1 \vee mbox2)$, from 5. and 7.
9. $(\neg mbox1 \wedge \neg mbox2) \rightarrow [save1](mbox1 \wedge (mbox1 \vee mbox2))$, from 1. and 8.
10. $(\neg mbox1 \wedge \neg mbox2) \rightarrow [save1](mbox1 \vee mbox2)$, from 9. and classical logic

■

This happens because no specific solution to the frame problem is associated to EPDL, and then, without considering the frame axiom $\neg mbox2 \rightarrow [save1]\neg mbox2$, we still get the above unintuitive result. As a way of avoiding to state frame axioms in the domain description, Foo and Zhang [120] suggest to generate them "on the fly", i.e., by the time queries are made. This could be achieved based on an interpolation result stating that the only frame axioms needed are those mentioning actions and atoms occurring in the vocabulary of the query. With this, according to the authors, it would be enough to use some method for automatically generating frame axioms from the effect laws, like, e.g. Pednault's [95].

The advantage of such an approach to the frame problem is the fact that no information about persistence has to be stated in the action theory. The inconvenience is that frame axioms are still needed and must be computed during the reasoning process. This constitutes an overhead that is neatly worse than that produced by checking the literal preservation condition of the dependence-based approach (cf. Section 4.4).

6.6 The Mailboxes Scenario with Dependences

So far we have seen the difficulties that arise when we try to formalize actions with both indeterminate and indirect effects in fluent-indexed causal approaches.

The problem with all these formalisms is that in our scenario there is an atom (*saved*) that can be caused in two different ways (directly with *save* or indirectly with *save*1 or *save*2) and that can or cannot cause nondeterministic ramifications depending on the way it was generated. With fluent-indexed approaches we cannot record this subtlety and this is the main reason they all fail in formalizing this example. So, with

all this discussion, we have seen that with the approaches presented in [78, 79, 83, 84, 112, 119, 35] either we get an exclusive interpretation of the nondeterminism, or we have to state frame axioms in the action theory.

Here we present the formalization of the Mailboxes Scenario in the dependence-based approach, which is action-indexed. The corresponding action theory for that is $\mathcal{D}_{mail} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$, where

$$\mathcal{T} = \left\{ \begin{array}{c} saved \leftrightarrow (mbox1 \vee mbox2), \\ [save]saved, [save1]mbox1, [save2]mbox2, \\ \langle save \rangle \top, \langle save1 \rangle \top, \langle save2 \rangle \top \end{array} \right\}$$

$$\rightsquigarrow = \left\{ \begin{array}{c} \langle save1, mbox1 \rangle, \langle save2, mbox2 \rangle, \\ \langle save1, saved \rangle, \langle save2, saved \rangle, \\ \langle save, saved \rangle, \langle save, mbox1 \rangle, \langle save, mbox2 \rangle \end{array} \right\}$$

Then, we have $\mathcal{D}_{mail} \models [save](mbox1 \vee mbox2)$, as intended.

This supports our thesis and others' [102, 101, 14] according to which causality must be action indexed, and also justifies our choice for the dependence-based solution to the frame problem. It is important to observe, however, that with it we do not entirely solve the ramification problem: while indirect effects such as $[save1]saved$ can be deduced with $\models_{\rightsquigarrow}$ without explicitly stating that in the set of laws for $save1$, we nevertheless still have to state *indirect dependences* such as $save1 \rightsquigarrow saved$. However, according to Reiter's view:

> "what counts as a solution to the frame problem ... is a systematic procedure for generating, from the effect laws, ... a parsimonious representation for [all] the frame axioms" [100].

The framework of $\rightsquigarrow$ complies with that as the dependence relation can be semi-automatically generated from the set of static and effect laws [13]. Moreover, as it has been shown in this chapter and argued in [15, 55], our approach is in line with the state of the art because none of the existing solutions to the frame and the ramification problems can handle domains with both indeterminate and indirect effects.

Chapter 7

where we open our toolbox

Refining Modularity and Computing Implicit Laws

So act that your principle of action might
safely be made a law for the whole world.

— Immanuel Kant

In this chapter, we make a step further into the concept of modularity. Besides considering a solution to the frame and ramification problems integrated in the base formalism, we develop a more fine grained analysis of modular theories. We achieve that by investigating some possible arrangements of modules and establishing a set of postulates that characterize modularity. Moreover, we also define algorithms to identify the troubled part of a given theory.

7.1 Defining Modules

Remembering our central hypothesis, what we argue for is that the different types of laws defined in Section 2.2 should be neatly separated in modules. Besides that, following the ideas in Chapter 4, we want such laws to interfere only in one sense: static laws together with action laws for a may have consequences that do not follow from the action laws for a alone (e.g. ramifications). The other way round, action laws should not allow to infer new static laws, action laws for a should not allow to infer action laws for a', etc. This means that our logical modules should be designed in such a way that they are as specialized and as little dependent on others as possible.

Our first claim is that the distinction made between the types of laws commonly used in reasoning about actions is not just a matter of syntactical sugar. By identifying such a distinction and treating different formulas in different ways, we tacitly assume that they constitute the basic entities in the theory of a domain description. It is not difficult to see why: to determine the set of possible states, static laws must be dealt with in a careful manner; in plan generation tasks, executabilities play an important role; for prediction and regression, the effect laws take their turn; etc.

To simplify the presentation, in this chapter we investigate how this can be accomplished when just one action is considered. A generalization of the results we obtain here is addressed in Chapter 8.

Given that, our first proposal here is to separate laws of different types into different pieces of a theory. Henceforth, the set of all static laws of a domain will be denoted by $\mathcal{S} \subseteq \mathfrak{Fml}$. For $a \in \mathfrak{Act}$, the set of effect laws for a is denoted by $\mathcal{E}^a$; the set of all executability laws for a will be denoted by $\mathcal{X}^a$; and all inexecutability laws for a is denoted by $\mathcal{I}^a$.

Definition 7.1 (Action theory for a)
An action theory for a *is a tuple* $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$, *where* $\mathcal{T} = \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a$.

In our running scenario example, an action theory for *tease* would comprise

$$\mathcal{S} = \{\mathit{walking} \rightarrow \mathit{alive}\},\ \mathcal{E}^{\mathit{tease}} = \{[\mathit{tease}]\mathit{walking}\},$$

$$\mathcal{X}^{\mathit{tease}} = \{\langle \mathit{tease} \rangle \top\},\ \mathcal{I}^{\mathit{tease}} = \{\neg \mathit{alive} \rightarrow [\mathit{tease}]\bot\},$$

and a dependence $\rightsquigarrow = \{\langle \mathit{tease}, \mathit{walking} \rangle\}$.

With these basic entities, we address now modularization of action theories. In what follows, given an action theory $\mathcal{D}^a$, we propose and analyze some possible arrangements of the sets $\mathcal{S}$, $\mathcal{E}^a$, $\mathcal{X}^a$ and $\mathcal{I}^a$ into what we call module prototypes (cf. Section 3.1). The purpose here is to argue backwards from analyzing what modules in reasoning about actions should be to a definition of modularity that better fits it. As we cannot cope with local completeness (cf. Section 3.3), what we do in the sequel is to relax such a principle and allow modules to have some degree of interaction. This will give us a "coupling-friendly" modularity [57].

Looking at the set $\mathcal{S}$ alone, we see that static laws do not mention actions at all, and then, in our context, they do not contain modal operators. This means that for inferences concerning only static laws, we need neither all expressiveness of PDL nor its

consequence relation. This suggests that static laws should constitute a module prototype in classical propositional logic. Let $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \rangle$ be such a sub-description, where $\mathcal{L}_{\mathsf{CPL}}$ is the language of the classical propositional logic, and $\models_{\mathsf{CPL}}$ is the classical entailment relation.

Regarding the solution to the frame and ramification problems, it could be reasonable to define a module prototype only for frame axioms. This would give us $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \emptyset \rangle$. Nevertheless, a solution to the frame problem is global to the theory, in the sense that its solution is necessary to most reasoning tasks. Moreover, deduction of frame axioms is mainly important in interacting with effect laws, and not for just deriving some frame axioms sporadically. Because of this we consider having a module like that would not really help modularity.

With a similar reasoning, we can expect to have a module prototype built on the effect laws $\mathcal{E}^a$ and $\models_{\mathsf{PDL}}$: $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{E}^a \rangle$, i.e., a sub-description for deriving effect laws. Unfortunately, in the presence of the frame and ramification problems, this is not enough: in all inferences about effect laws, information about frame axioms and indirect effects (ruled by a causal notion) must be taken into account. This means that $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{E}^a \rangle$ would not be good as a module for reasoning about actions.

By applying the same analysis as in the above paragraphs, we can see that $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{X}^a \rangle$ and $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{I}^a \rangle$ are not good as modules either. Even if, *a priori*, we do not need frame axioms to infer executabilities, the absence of static laws is too restrictive. For the case of inexecutabilities, as long as they can be seen as a special type of effect laws, frame axioms are important, or, as we are going to see in the sequel, because of some overlaps between $\mathcal{E}^a$ and $\mathcal{I}^a$, we should at least guarantee that all inexecutabilities entailed by the theory are in $\mathcal{I}^a$ (and thus $\mathcal{S}$ is mandatory).

Because static laws describe the laws of the universe being represented (and that must be respected in every reasoning), it is reasonable to consider them as part of every module.[1] One of the reasons for that is the situation illustrated above: without $\mathcal{S}$, it is not possible to derive indirect effects with $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{E}^a \rangle$. The same can be said about the solution to the frame problem: it should be present at least when effects are under concern. Moreover, there are trivial effect laws that are entailed by $\mathcal{I}^a$: $\varphi \rightarrow [a]\bot$ entails $\varphi \rightarrow [a]\psi$ for any $\psi \in \mathfrak{Fml}$.[2] In this case, we may also need inex-

[1] We could also see them as global data with a special status, similarly as done in [64]. For the sake of presentation, we prefer to keep static laws in the same level as action laws, i.e., seeing them just as formulas of a theory, so that the difference is just what they are for. Of course, in real implementations there should be no redundant replications of the set $\mathcal{S}$.

[2] If we were to argue against the principle of explosion (cf. Section 3.3), this could be a reason.

ecutabilities to guarantee the module's completeness. So a module prototype better than $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{E}^a \rangle$ would rather be $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle$. With it, all effects, non-effects and ramifications of actions should be derived.

As long as action laws other than elements of $\mathcal{X}^a$ are not necessarily needed to infer executabilities, we can expect $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{X}^a \rangle$ to be a module prototype for deriving executability laws. Similarly, and despite the fact that $\mathcal{E}^a$ plays a role in the deduction of inexecutabilities, we shall define $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle$ as a module prototype for inexecutability laws.

So, now we have four module prototypes: one for inferring in classical logic, $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \rangle$; one for doing prediction and explanation in PDL with a solution to the frame and ramification problems, $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle$; a module prototype for inferring executability laws $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{X}^a \rangle$; and one for the deduction of inexecutabilities, $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle$. Such sub-descriptions are minimal in the sense that each one contains the minimum necessary potential interaction inside their data to the realization of inferences in its domain of application. For instance, as argued above, weakening $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle$ may have as consequence that some laws will no longer be inferable in the module.

With that, we define our version of local completeness that gives us modularity:

Definition 7.2 (a-modularity)
Let $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$ be an action theory for a such that $\mathcal{T} = \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a$. $\mathcal{D}^a$ is a-modular *if and only if*

1. $\mathcal{D}^a \models \varphi$ *implies* $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \rangle \models \varphi$
2. $\mathcal{D}^a \models \varphi \rightarrow \langle a \rangle \top$ *implies* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{X}^a \rangle \models \varphi \rightarrow \langle a \rangle \top$
3. $\mathcal{D}^a \models \varphi \rightarrow [a] \bot$ *implies* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle \models \varphi \rightarrow [a] \bot$
4. $\mathcal{D}^a \models \varphi \rightarrow [a] \psi$ *implies* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle \models \varphi \rightarrow [a] \psi$

The main difference between our definition of local completeness and those of Garson and Cuenca Grau *et al.* (cf. Section 3.3) is that we do not require modules to be disjoint modulo logical consequences. In other words, we allow for a formula of a given type to be inferred from different modules.

Just having module prototypes defined in our way is not enough to have a-modularity. This is what we address in the sequel.

7.2 More Fine Grained Postulates

A first step toward modularity has been the proposed division of our entities into modules. Recalling the discussion in Chapter 3, in order to accomplish our goal, we have to diminish interaction among such modules, rendering them the least interwoven we can.

Restricted to the case of one action, in the rest of this chapter we will state and investigate postulates that guarantee modularity, and give a method to satisfy them. Although we here use the syntax of PDL, all we shall say applies as well to first-order formalisms, in particular to the Situation Calculus. All postulates we are going to present can be stated as well for other frameworks, in particular for action languages such as $\mathcal{A}$, $\mathcal{AR}$ [39, 65, 44] and others, and for Situation Calculus based approaches. In [57] we have given a Situation Calculus version of our analysis.

Let $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\sim}, \mathcal{T} \rangle$ be such that $\mathcal{T} = \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a$.

PC (**Logical consistency**): $\mathcal{D}^a \not\models \bot$

The theory of a given action should be logically consistent.

PS (**No implicit static laws**): if $\mathcal{D}^a \models \varphi$, then $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \rangle \models \varphi$

If a classical formula can be inferred from the action theory, then it should be inferable from the set of static laws alone.

PI (**No implicit inexecutability laws**):

$$\text{if } \mathcal{D}^a \models \varphi \rightarrow [a]\bot, \text{ then } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle \models \varphi \rightarrow [a]\bot$$

If an inexecutability law for a given action a can be inferred from its domain description, then it should be inferable in PDL from the static laws and the set of inexecutability laws for a alone.

PX (**No implicit executability laws**):

$$\text{if } \mathcal{D}^a \models \varphi \rightarrow \langle a \rangle \top, \text{ then } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{X}^a \rangle \models \varphi \rightarrow \langle a \rangle \top$$

If an executability law for a can be inferred from its action theory, then it should already "be" in $\mathcal{X}^a$, in the sense that it should also be inferable in PDL from the set of static and executability laws for a alone.

Postulate **PC** is obvious, for we are interested in consistent theories. Moreover, it can be shown that **PX** is a consequence of **PS** (see Corollary 8.2).

Thus, while **PC** is obvious and **PX** can be ensured by **PS**, things are less obvious for Postulates **PS** and **PI**: it turns out that, for all approaches in the literature, they are easily violated by action theories that allow to express the four kinds of laws. We therefore study each of these postulates in the subsequent sections by means of examples, give algorithms to decide whether they are satisfied, and discuss about what to do in the case the answer is "no".

7.3 No Implicit Static Laws

While executability laws increase expressive power, they might conflict with inexecutability laws. Consider, for example, $\mathcal{D}^{tease}_{wts} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{tease} \cup \mathcal{X}^{tease} \cup \mathcal{I}^{tease} \rangle$, where

$$\mathcal{S} = \{walking \rightarrow alive\},\ \mathcal{E}^{tease} = \{[tease]walking\},$$

$$\mathcal{X}^{tease} = \{\langle tease \rangle \top\},\ \mathcal{I}^{tease} = \{\neg alive \rightarrow [tease]\bot\}$$

and the dependence relation is given by $\rightsquigarrow = \{\langle tease, walking \rangle\}$.

From this description, we have the unintuitive inference $\mathcal{X}^{tease}, \mathcal{I}^{tease} \models_{\mathsf{PDL}} alive$: the turkey is immortal (Figure 7.1)! This is an implicit static law (cf. Section 4.3) because *alive* does not follow from $\mathcal{S}$ alone: $\mathcal{D}^{tease}_{wts}$ violates Postulate **PS**.

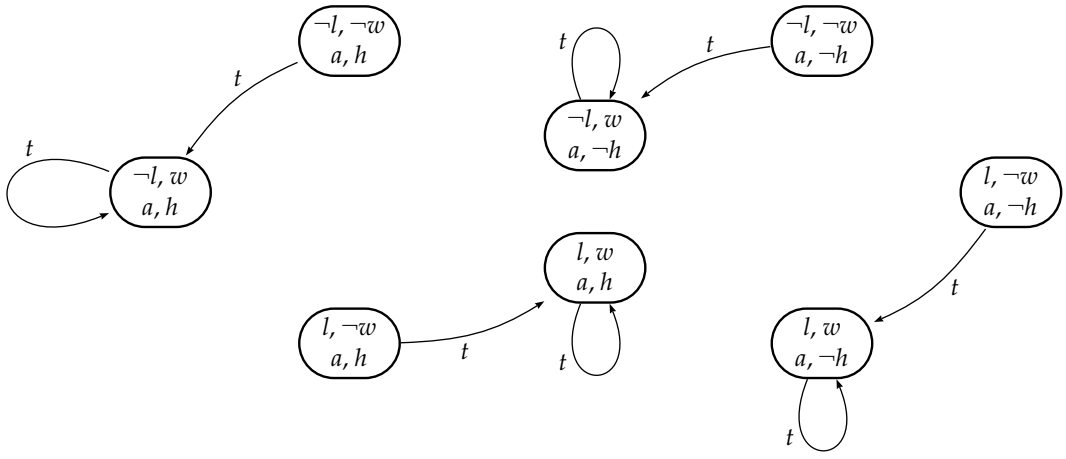

Figure 7.1: A $\rightsquigarrow$-model for the theory in $\mathcal{D}^{tease}$: the turkey is immortal.

Implicit static laws are not a drawback of our underlying logical formalism. They also appear in Situation Calculus-based approaches and in causal laws theories. To

witness[3], suppose in Lin's framework we have

$$Holds(p_1, s) \rightarrow Caused(p_2, true, s) \tag{7.1}$$

and

$$Caused(p_2, false, s) \tag{7.2}$$

Then from (7.2) and Axiom (6.2), we get

$$\neg Holds(p_2, s) \tag{7.3}$$

From (7.2) and the contrapositive of Axiom (6.1) it follows

$$\neg Caused(p_2, true, s) \tag{7.4}$$

Finally, from (7.1) and (7.4) we get

$$\neg Holds(p_1, s)$$

which is an implicit static law.

To see how implicit static laws show up in McCain and Turner's causal laws approach (cf. Section 6.3), let **Laws** contain the causal law $\varphi \Rightarrow \psi$ and $\mathcal{T} = \{\neg\psi\}$. Then $\neg\varphi$ is an implicit static law in such a description.

How can we find out whether an action theory for a satisfies Postulate **PS**? Before that, we need a definition.

Definition 7.3 (Big model)
Let $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$ *be such that* $\mathcal{T} = \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a$. *Then* $\mathscr{M} = \langle W, R \rangle$ *is the* big *(alias* maximal/standard*)* model *for* $\mathcal{D}^a$ *if and only if:*

- $\mathscr{M}$ *is a* $\rightsquigarrow$*-model;*
- $W = valuations(\mathcal{S})$ *(all valuations of* $\mathcal{S}$*); and*
- $R_a = \{(w, w') : \text{ for all } \varphi \rightarrow [a]\psi \in \mathcal{E}^a \cup \mathcal{I}^a, \text{ if } \models^{\mathscr{M}}_{w} \varphi, \text{ then } \models^{\mathscr{M}}_{w'} \psi\}$.

For an example, consider an action theory whose components are given by

$$\mathcal{S} = \emptyset,\ \mathcal{E}^a = \{p_1 \rightarrow [a]\neg p_2\},\ \mathcal{X}^a = \{\langle a \rangle \top\},$$

[3]The examples are from [104].

$$\mathcal{I}^a = \{p_2 \to [a]\bot\}, \text{ and } \rightsquigarrow = \{\langle a, \neg p_1\rangle, \langle a, \neg p_2\rangle\}$$

Figure 7.2 depicts one of its models and its associated big model.

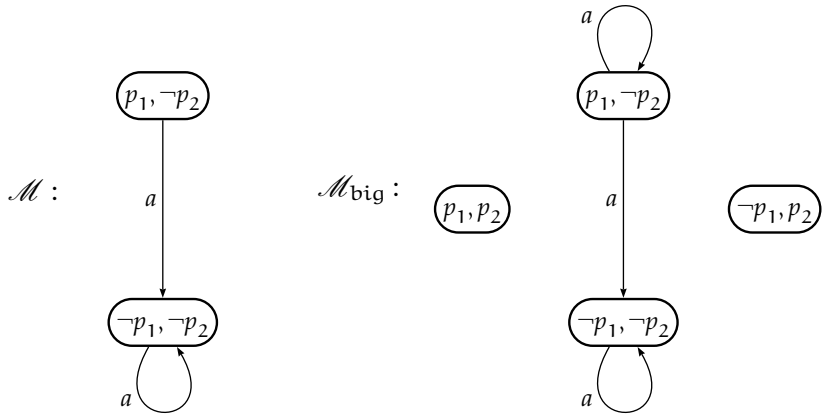

Figure 7.2: A model of $\mathcal{D}^a$ and the big model $\mathscr{M}_{\mathtt{big}}$ of $\mathcal{D}^a$.

Big models contain all valuations consistent with $\mathcal{S}$. Clearly, for a big model $\mathscr{M}$ we have $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{I}^a$. Because $\mathscr{M}$ extends the set of possible worlds, it is only $\mathcal{X}^a$ which might not be true in $\mathscr{M}$.

Theorem 7.1
Let $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T}\rangle$ be such that $\mathcal{T} = \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a$. $\mathcal{D}^a$ satisfies Postulate **PS** *if and only if the big model for $\mathcal{D}^a$ is a model of $\mathcal{D}^a$.*

Proof:
Let $\mathscr{M} = \langle W, R\rangle$ be the big model of $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a\rangle$.

($\Rightarrow$): As $\mathscr{M}$ is a big model of $\mathcal{D}^a$, we have $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{I}^a$. It remains to show that $\models^{\mathscr{M}} \mathcal{X}^a$. Let $\varphi_i \to \langle a\rangle\top \in \mathcal{X}^a$, and let $w \in W$ be such that $\models^{\mathscr{M}}_w \varphi_i$. Therefore, for all $\varphi_j \in \mathfrak{Fml}$ such that $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi_j \to [a]\bot$, we must have $\not\models^{\mathscr{M}}_w \varphi_j$, because $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\rightsquigarrow} \neg(\varphi_i \wedge \varphi_j)$, and as $\mathcal{D}^a$ satisfies Postulate **PS**, $\mathcal{S} \models_{\mathsf{CPL}} \neg(\varphi_i \wedge \varphi_j)$, and hence $\models^{\mathscr{M}} \neg(\varphi_i \wedge \varphi_j)$. Then, by the construction of $\mathscr{M}$, there is some $w' \in W$ such that $\models^{\mathscr{M}}_{w'} \psi$, for all $\varphi \to [a]\psi$ such that $\mathcal{S}, \mathcal{E}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi \to [a]\psi$ and $\models^{\mathscr{M}}_w \varphi$, and wR_aw'. Hence, $\models^{\mathscr{M}}_w \varphi_i \to \langle a\rangle\top$, and thus $\mathscr{M}$ is a model of $\mathcal{D}^a$.

($\Leftarrow$): Suppose $\mathcal{D}^a$ does not satisfy Postulate **PS**. Then there must be $\varphi \in \mathfrak{Fml}$ such that $\mathcal{D}^a \models \varphi$ and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S}\rangle \not\models \varphi$, i.e., $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi$ and $\mathcal{S} \not\models_{\mathsf{CPL}} \varphi$. Hence there is a valuation *val* of $\mathcal{S}$ that falsifies φ. As $val \in W$ (because $\mathscr{M}$ contains all possible valuations of $\mathcal{S}$), $\mathscr{M}$ is not a model of $\mathcal{D}^a$. ■

In the rest of this section, we will characterize when a domain description admits a big model.

We shall give an algorithm to find a finite characterization of all[4] implicit static laws of a given action theory $\mathcal{D}^a$. The idea follows that of Algorithm 4.1 with the improvement of taking into account dependence information: for each executability law $\varphi \rightarrow \langle a \rangle \top$ in the theory, construct from $\mathcal{E}^a$, $\mathcal{I}^a$ and $\rightsquigarrow$ a set of inexecutabilities $\{\varphi_1 \rightarrow [a]\bot, \ldots, \varphi_n \rightarrow [a]\bot\}$ that potentially conflict with $\varphi \rightarrow \langle a \rangle \top$. For each i, $1 \leq i \leq n$, if $\varphi \wedge \varphi_i$ is satisfiable w.r.t. $\mathcal{S}$, mark $\neg(\varphi \wedge \varphi_i)$ as an implicit static law. In the same way as done in Algorithm 4.2, incrementally repeat this procedure (adding all the implicit $\neg(\varphi \wedge \varphi_i)$ to $\mathcal{S}$) until no more implicit static law is obtained.

For an example of the execution of the algorithm, consider the action theory for *tease* above. For the action *tease*, we have the executability $\langle \mathit{tease} \rangle \top$. Now, from $\mathcal{E}^{\mathit{tease}}$, $\mathcal{I}^{\mathit{tease}}$ and $\rightsquigarrow$, we try to build an inexecutability for *tease*. We take $[\mathit{tease}]\mathit{walking}$ and compute then all indirect effects of *tease* w.r.t. $\mathcal{S}$. From $\mathit{walking} \rightarrow \mathit{alive}$, we get that *alive* is an indirect effect of *tease*, giving us $[\mathit{tease}]\mathit{alive}$. But $\langle \mathit{tease}, \mathit{alive} \rangle \notin \rightsquigarrow$, which means the frame axiom $\neg\mathit{alive} \rightarrow [\mathit{tease}]\neg\mathit{alive}$ holds. Together with $[\mathit{tease}]\mathit{alive}$, this gives us the inexecutability $\neg\mathit{alive} \rightarrow [\mathit{tease}]\bot$. As $\mathcal{S} \cup \{\top, \neg\mathit{alive}\}$ is satisfiable ($\top$ is the antecedent of the executability $\langle \mathit{tease} \rangle \top$), we get $\neg\mathit{alive} \rightarrow \bot$, i.e., the implicit static law *alive*. For this example, no other inexecutability for *tease* can be derived, so the computation stops.

Before presenting the pseudo-code of the algorithm, we need some definitions.

Definition 7.4 (Implicate)
Let $\varphi \in \mathfrak{Fml}$ and χ be a clause. χ is an implicate of φ *if and only if $\varphi \models_{\overline{\mathsf{CPL}}} \chi$.*

In our running example, $\mathit{walking} \vee \mathit{alive}$ and $\neg\mathit{walking} \vee \mathit{alive}$ are implicates of the set of formulas $\{\mathit{walking} \rightarrow \mathit{alive}, \mathit{walking}\}$.

Definition 7.5 (Prime implicate)
Let $\varphi \in \mathfrak{Fml}$ and χ be a clause. χ is a prime implicate of φ *if and only if*

- *χ is an implicate of φ, and*
- *for every implicate χ' of φ, $\chi' \models_{\overline{\mathsf{CPL}}} \chi$ implies $\chi \models_{\overline{\mathsf{CPL}}} \chi'$.*

The set of all prime implicates of a formula φ is denoted $PI(\varphi)$.

[4] Actually, what the algorithm does is to find an interpolant of all implicit static laws of the theory.

For example, the set of prime implicates of p_1 is just $\{p_1\}$, and that of $p_1 \wedge (\neg p_1 \vee p_2) \wedge (\neg p_1 \vee p_3 \vee p_4)$ is $\{p_1, p_2, p_3 \vee p_4\}$. In our shooting domain, *alive* is a prime implicate of $\{walking \rightarrow alive, walking\}$. For more on prime implicates and their properties, see [82].

Definition 7.6 (Function *NewCons*(.))
Let $\varphi, \psi \in \mathfrak{Fml}$. *Then* $NewCons(\psi, \varphi) = PI(\varphi \wedge \psi) \setminus PI(\varphi)$.

The function $NewCons(\psi, \varphi)$ computes the *new consequences* of ψ w.r.t. φ: the set of strongest clauses that follow from $\varphi \wedge \psi$, but do not follow from φ alone (cf. e.g. [61]). It is computed by subtracting the prime implicates of φ from those of $\varphi \wedge \psi$. For example, $NewCons((\neg p_1 \vee p_2) \wedge (\neg p_1 \vee p_3 \vee p_4), p_1) = \{p_2, p_3 \vee p_4\}$. And for our scenario, $NewCons(walking, walking \rightarrow alive) = \{alive, walking\}$.

The algorithm below improves both Algorithms 4.1 and 4.2 by integrating a solution to the frame problem (via the dependence relation $\rightsquigarrow$). For convenience, we define $\mathcal{C}^a = \mathcal{E}^a \cup \mathcal{I}^a$ as the set of all formulas expressing the direct consequences of an action a, whether they are consistent or not.

Algorithm 7.1 Finding all implicit static laws induced by a

input: $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$
output: $\mathcal{S}_{imp^*}$, the set of all implicit static laws of $\mathcal{D}^a$

```
S_imp* := ∅
C^a := E^a ∪ I^a
repeat
  S_imp := ∅
  for all φ → ⟨a⟩⊤ ∈ X^a do
    for all Ĉ^a ⊆ C^a such that Ĉ^a ≠ ∅ do
      φ_Ĉ^a := ⋀{φ_i : φ_i → [a]ψ_i ∈ Ĉ^a}
      ψ_Ĉ^a := ⋀{ψ_i : φ_i → [a]ψ_i ∈ Ĉ^a}
      for all χ ∈ NewCons(ψ_Ĉ^a, S) do
        if S ∪ S_imp* ∪ {φ, φ_Ĉ^a, ¬χ} ⊭_CPL ⊥ and ∀ℓ_i ∈ χ, a ↛ ℓ_i then
          S_imp := S_imp ∪ {¬(φ ∧ φ_Ĉ^a ∧ ¬χ)}
  S_imp* := S_imp* ∪ S_imp
until S_imp = ∅
```

In each step of the algorithm, $\mathcal{S} \cup \mathcal{S}_{imp^*}$ is the updated set of static laws (the original ones fed with the implicit laws caught up to that point). At the end, $\mathcal{S}_{imp^*}$ collects all the implicit static laws.

The following result establishes decidability of the method:

Theorem 7.2 (Decidability)
Algorithm 7.1 terminates.

Proof:
Let $\mathcal{C}^a = \mathcal{E}^a \cup \mathcal{I}^a$. First, the set of candidates to be an implicit static law that might be due to a and that are examined in the **repeat**-loop is

$$\{\neg(\varphi \wedge \varphi_{\hat{\mathcal{C}}^a} \wedge \neg\chi) \,:\, \hat{\mathcal{C}}^a \subseteq \mathcal{C}^a, \varphi \rightarrow \langle a\rangle\top \in \mathcal{X}^a \text{ and } \chi \in \mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S})\}$$

As $\mathcal{E}^a$, $\mathcal{I}^a$ and $\mathcal{X}^a$ are finite, this set is finite.

In each step, either the algorithm stops because $\mathcal{S}_{imp} = \emptyset$, or at least one of the candidates is put into $\mathcal{S}_{imp}$ in the outermost **for**-loop. (This one terminates, because $\mathcal{X}^a$, $\mathcal{C}^a$ and $\mathit{NewCons}(.)$ are finite.) Such a candidate is not going to be put into $\mathcal{S}_{imp}$ in future steps, because once added to $\mathcal{S} \cup \mathcal{S}_{imp^*}$, it will be in the set of laws $\mathcal{S} \cup \mathcal{S}_{imp^*}$ of all subsequent executions of the outermost **for**-loop, falsifying its respective **if**-test for such a candidate. Hence the **repeat**-loop is bounded by the number of candidates, and therefore Algorithm 7.1 terminates. ■

While terminating, our algorithm comes with considerable computational costs: first, the number of formulas $\varphi_{\hat{\mathcal{C}}^a}$ and $\psi_{\hat{\mathcal{C}}^a}$ is exponential in the size of $\mathcal{C}^a$, and second, the computation of $\mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S})$ might result in exponential growth. While we might expect $\mathcal{C}^a$ to be reasonably small in practice (because $\mathcal{E}^a$ and $\mathcal{I}^a$ are in general small), the size of $\mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S})$ is more difficult to control.

Example 7.1
For $\mathcal{D}^{tease}_{wts}$, Algorithm 7.1 returns $\mathcal{S}_{imp^*} = \{alive\}$.

The following theorem establishes soundness and completeness of our method:

Theorem 7.3
Let $\mathcal{S}_{imp^*}$ be the output of Algorithm 7.1 on input $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a\rangle$. Then $\mathcal{D}^a$ satisfies Postulate **PS** *if and only if $\mathcal{S}_{imp^*} = \emptyset$.*

Proof:
See Appendix C. ■

Corollary 7.1
Let $\mathcal{S}_{imp^*}$ be the output of Algorithm 7.1 on input $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a\rangle$. Then

1. *$\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{S}_{imp^*} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a\rangle$ satisfies* **PS**.
2. *$\mathcal{D}^a \models \bigwedge \mathcal{S}_{imp^*}$.*

Proof:
Item 1. is straightforward from the termination of Algorithm 7.1 and Theorem 7.3. Item 2. follows from the fact that by the **if**-test in Algorithm 7.1, the only formulas that are put in $\mathcal{S}_{imp^*}$ at each execution of the **repeat**-loop are exactly those that are implicit static laws of the current theory, and therefore of the original theory, too. ■

Corollary 7.2
For all $\varphi \in \mathfrak{Fml}$, $\mathcal{D}^a \models \varphi$ *if and only if* $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \cup \mathcal{S}_{imp^*} \rangle \models \varphi$.

Proof:
For the left-to-right direction, let $\varphi \in \mathfrak{Fml}$ be such that $\mathcal{D}^a \models \varphi$, i.e., $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\sim\!\!>} \varphi$, and hence $\mathcal{S} \cup \mathcal{S}_{imp^*}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\sim\!\!>} \varphi$, by monotonicity. By Corollary 7.1-1., we have that $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\sim\!\!>}, \mathcal{S} \cup \mathcal{S}_{imp^*} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$ has no implicit static law. From this it follows $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \cup \mathcal{S}_{imp^*} \rangle \models \varphi$.
The right-to-left direction is straightforward by Corollary 7.1-2. ■

What shall we do once we have discovered an implicit static law?

The presence of implicit static laws may indicate too strong executability laws: in Example 7.1, we wrongly assumed that *tease* is always executable. Thus one way of "repairing" our theory would be to consider the weaker executability $alive \rightarrow \langle tease \rangle \top$ instead of $\langle tease \rangle \top$ in $\mathcal{X}^{tease}$.

On the other hand, implicit static laws may also indicate that the inexecutability laws are too strong:

Example 7.2
Consider $\mathcal{D}^{shoot}_{wts}$ such that $\mathcal{S} = \emptyset$, $\mathcal{E}^{shoot} = \{loaded \rightarrow [shoot]\neg alive\}$, $\mathcal{X}^{shoot} = \{hasGun \rightarrow \langle shoot \rangle \top\}$ and $\mathcal{I}^{shoot} = \{[shoot]\bot\}$, with $\rightsquigarrow = \{\langle shoot, \neg alive \rangle, \langle shoot, \neg walking \rangle$. For this action theory, Algorithm 7.1 returns $\mathcal{S}_{imp^*} = \{\neg hasGun\}$.

In Example 7.2, we discovered that the agent never has a gun. The problem here can be overcome by weakening $[shoot]\bot$ in $\mathcal{I}^{shoot}$ with $\neg hasGun \rightarrow [shoot]\bot$.[5]

We can go further on in this reasoning and also argue that the problem may be due to a too strong set of effect laws, or even to too strong frame axioms (i.e., a too weak dependence relation). To witness, for Example 7.1, if we take off the inexecutability $\neg alive \rightarrow [tease]\bot$ and replace the law $[tease]walking$ by the weaker $alive \rightarrow [tease]walking$, the resulting action theory would satisfy Postulate **PS**. In the

[5]Regarding Examples 7.1 and 7.2, one might argue that in practice such silly errors will never be made. Nevertheless, the examples here given are quite simplistic, and for applications of real interest, whose complexity will be much higher, we simply cannot rely on the designer's knowledge about all side effects the stated formulas can have.

same way, stating the (unintuitive) dependence $tease \rightsquigarrow alive$ (which means the frame axiom $\neg alive \rightarrow [tease]\neg alive$ is no longer valid) guarantees satisfaction of **PS**. (Note, however, that this solution becomes intuitive when *alive* is replaced by *awake*.)

To finish, implicit static laws of course may also indicate that the static laws themselves are too weak:

Example 7.3
Suppose a computer representation of the line of integers, in which we can be at a strictly positive number, *positive*, or at a negative one or zero, $\neg positive$. Let *maxInt* and *minInt*, respectively, be the largest and the smallest representable integer number. Action *goLeft* is the action of moving to the biggest integer strictly smaller than the one at which we are. Consider the action theory $\mathcal{D}_{\mathbb{Z}}^{goLeft}$ for this scenario such that ($at_{\mathtt{i}}$ means we are at number $\mathtt{i}$):

$$\mathcal{S} = \{at_{\mathtt{i}} \rightarrow positive \ : \ 0 < \mathtt{i} \leq maxInt\} \cup \{at_{\mathtt{i}} \rightarrow \neg positive \ : \ minInt \leq \mathtt{i} \leq 0\}$$

$$\mathcal{E}^{goLeft} = \{at_{minInt} \rightarrow [goLeft]underflow\} \cup \{at_{\mathtt{i}} \rightarrow [goLeft]at_{\mathtt{i}-1} \ : \ \mathtt{i} > minInt\},$$

$$\mathcal{X}^{goLeft} = \{\langle goLeft\rangle \top\}, \ \mathcal{I}^{goLeft} = \emptyset$$

with the dependence relation ($minInt \leq \mathtt{i} < maxInt$):

$$\rightsquigarrow = \left\{ \begin{array}{c} \langle goLeft, at_{\mathtt{i}}\rangle, \langle goLeft, positive\rangle, \\ \langle goLeft, \neg positive\rangle, \langle goLeft, underflow\rangle \end{array} \right\}$$

Applying Algorithm 7.1 to this action theory would give us the implicit static law $\neg(at_1 \wedge at_2)$, i.e., we cannot be at numbers 1 and 2 at the same time.

To summarize, in order to satisfy Postulate **PS**, an action theory should contain a complete set of static laws or, alternatively, should not contain too strong action laws (executability, inexecutability or effect laws). We will come back to this point in Chapter 9, where we address action theory change.

Remark 7.1 $\mathcal{S} \cup \mathcal{S}_{imp^*}$ in general is not intuitive.

Whereas in the latter example the implicit static laws should be added to $\mathcal{S}$, in the others the implicit static laws are unintuitive and due to an (in)executability law that is too strong and should be weakened. Of course, how intuitive the modified action theory will be depends mainly on the knowledge engineer's choice.

7.4 No Implicit Inexecutability Laws

Let $\mathcal{D}_{wts}^{tease}$ be such that $\mathcal{S} = \{walking \rightarrow alive\}$, $\mathcal{E}^{tease} = \{[tease]walking\}$, $\mathcal{X}^{tease} = \mathcal{I}^{tease} = \emptyset$, and $\rightsquigarrow = \{\langle tease, walking\rangle\}$. $\mathcal{D}_{wts}^{tease}$ in this way satisfies Postulate **PS**. Now we observe that from $[tease]walking$ it follows with $\mathcal{S}$ that $[tease]alive$, i.e., in every situation, after teasing the turkey, it is alive: $\mathcal{D}_{wts}^{tease} \models [tease]alive$. Now as $tease \not\rightsquigarrow alive$, the status of *alive* is not modified by *tease*, and we have $\mathcal{D}_{wts}^{tease} \models \neg alive \rightarrow [tease]\neg alive$. From the above, it follows

$$\mathcal{D}_{wts}^{tease} \models \neg alive \rightarrow [tease]\bot,$$

i.e., an inexecutability law stating that a dead turkey cannot be teased. But

$$\mathcal{S}, \mathcal{I}^{tease} \not\models_{\mathsf{PDL}} \neg alive \rightarrow [tease]\bot,$$

and then

$$\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^{tease} \rangle \not\models \neg alive \rightarrow [tease]\bot$$

which means that Postulate **PI** is violated. Here the formula $\neg alive \rightarrow [tease]\bot$ is an example of what we call an *implicit inexecutability law*.

In the literature, such laws are also known as *implicit qualifications* [42], and it has been often supposed, in a more or less tacit way, that it is a positive feature of frameworks to leave them implicit and provide mechanisms for inferring them [78, 79, 113]. The other way round, one might argue as well that implicit qualifications indicate that the domain has not been described in an adequate manner: the form of inexecutability laws is simpler than that of effect laws, and it might be reasonably expected that it is easier to exhaustively describe them.[6] Thus, all inexecutabilities of a given action should be explicitly stated, and this is what Postulate **PI** says.

How can we check whether **PI** is violated? We can conceive an algorithm to find implicit inexecutability laws of a given action a. The basic idea is as follows: for every combination of effect laws of the form $(\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow [a](\psi_1 \wedge \ldots \wedge \psi_n)$, with each $\varphi_i \rightarrow [a]\psi_i \in \mathcal{E}^a$, if $\varphi_1 \wedge \ldots \wedge \varphi_n$ is consistent w.r.t. to $\mathcal{S}$, $\psi_1 \wedge \ldots \wedge \psi_n$ inconsistent w.r.t. $\mathcal{S}$, and $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} (\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow [a]\bot$, then output $(\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow [a]\bot$ as an implicit inexecutability law. Our algorithm basically does this, and moreover takes into account dependence information.

[6] Note that this concerns the necessary conditions for executability, and thus it is not related to the qualification problem [85], which basically says that it is difficult to state all the sufficient conditions for executability of an action.

For an example of the execution of the algorithm, take $\mathcal{D}^{tease}_{wts}$ as above. From $\mathcal{E}^{tease}$ we get $\top \rightarrow [tease]walking$, whose antecedent is consistent with $\mathcal{S}$. As $\models_{\rightsquigarrow} \neg alive \rightarrow [tease]\neg alive$ and $\mathcal{S} \cup \{walking\} \models_{\overline{\mathsf{CPL}}} alive$, and because $\mathcal{S}, \mathcal{I}^{tease} \not\models_{\mathsf{PDL}} (\top \wedge \neg alive) \rightarrow [tease]\bot$, we caught an implicit inexecutability. As there is no other combination of effect laws for *tease*, we end the simulation here.

Algorithm 7.2 below shows the pseudo-code for that (the reason $\mathcal{X}^a$ is not used in the computation will be made clear in the sequel).

Algorithm 7.2 Finding implicit inexecutability laws for a

input: $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$
output: $\mathcal{I}^a_{imp}$, the set of implicit inexecutability laws for a

```
I^a_imp := ∅
for all Ê^a ⊆ E^a do
    φ_Ê^a := ⋀{φ_i : φ_i → [a]ψ_i ∈ Ê^a}
    ψ_Ê^a := ⋀{ψ_i : φ_i → [a]ψ_i ∈ Ê^a}
    for all χ ∈ NewCons(ψ_Ê^a, S) do
        if ∀ℓ_i ∈ χ, a ↛ ℓ_i and S, I^a ⊭_PDL (φ_Ê^a ∧ ¬χ) → [a]⊥ then
            I^a_imp := I^a_imp ∪ {(φ_Ê^a ∧ ¬χ) → [a]⊥}
```

Theorem 7.4 (Decidability)
Algorithm 7.2 terminates.

Proof:
Straightforward, as we have assumed $\mathcal{S}$, $\mathcal{E}^a$, $\mathcal{X}^a$, $\mathcal{I}^a$ and $\rightsquigarrow$ finite, and $NewCons(.)$ is finite (because $\mathcal{S}$ and $\psi_{\hat{\mathcal{E}}^a}$ are finite). ■

Example 7.4
Consider $\mathcal{D}^{tease}_{wts}$ as given above. Then Algorithm 7.2 returns $\mathcal{I}^{tease}_{imp} = \{\neg alive \rightarrow [tease]\bot\}$.

Nevertheless, applying Algorithm 7.2 is not enough to guarantee Postulate **PI**, as illustrated by the following example:

Example 7.5 (Incompleteness of Algorithm 7.2 without PS)
Let $\mathcal{D}^a$ be such that $\mathcal{S} = \emptyset$, $\mathcal{E}^a = \{p_1 \rightarrow [a]p_2\}$, $\mathcal{X}^a = \{\langle a \rangle \top\}$, $\mathcal{I}^a = \{p_2 \rightarrow [a]\bot\}$, and $\rightsquigarrow = \emptyset$. Then we have $\mathcal{D}^a \models p_1 \rightarrow [a]\bot$, but after running Algorithm 7.2 on $\mathcal{D}^a$ we have $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \cup \mathcal{I}^a_{imp} \rangle \not\models p_1 \rightarrow [a]\bot$.

Example 7.5 shows that the presence of implicit static laws (induced by executabilities) implies the existence of implicit inexecutabilities that are not caught by Algo-

rithm 7.2. One possibility of getting rid of this is by considering the weaker version of Postulate **PI**:

PI′ (**No implicit inexecutability laws – weak version**):

$$\text{if } \mathcal{D}^a \models \varphi \rightarrow [a]\bot \text{ and } \mathcal{D}^a \not\models \neg\varphi, \text{ then } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle \models \varphi \rightarrow [a]\bot$$

If a non-trivial inexecutability law for a given action a can be inferred from its respective theory, then it should be inferable in PDL from the static and inexecutability laws for it alone.

With an adaptation of Algorithm 7.2 to support a test for satisfiability of an inexecutability's antecedent, we could guarantee completeness with respect to Postulate **PI′**. However, such a test has the same complexity as checking whether Postulate **PS** is satisfied. That is the reason we keep abide on **PI** and require $\mathcal{D}^a$ to satisfy Postulate **PS** prior to running Algorithm 7.2. This gives us the following result:

Theorem 7.5
Let $\mathcal{I}^a_{imp}$ be the output of Algorithm 7.2 on input $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$. If $\mathcal{D}^a$ satisfies Postulate **PS**, *then $\mathcal{D}^a$ satisfies Postulate* **PI** *if and only if $\mathcal{I}^a_{imp} = \emptyset$.*

Proof:
See Appendix C. ■

With Algorithm 7.2, not only do we decide whether Postulate **PI** is satisfied, but we also get information on how to "repair" the action theory. The set of implicit inexecutabilities so obtained provides logical and meta-logical information concerning the correction that must be carried out: in the first case, elements of $\mathcal{I}^a_{imp}$ can be added to $\mathcal{I}^a$; in the second one, $\mathcal{I}^a_{imp}$ helps in properly changing $\mathcal{E}^a$ or $\rightsquigarrow$. For instance, to correct the action theory of our example, the knowledge engineer would have the following options:

1. Add the qualification $\neg alive \rightarrow [tease]\bot$ to $\mathcal{I}^{tease}$; or
2. Add the (unintuitive) dependence $\langle tease, alive \rangle$ to $\rightsquigarrow$; or
3. Weaken the effect law $[tease]walking$ to $alive \rightarrow [tease]walking$ in $\mathcal{E}^{tease}$.

It is easy to see that whatever she opts for, the resulting action theory for *tease* will satisfy Postulate **PI** (while still satisfying **PS**).

Example 7.6 (Drinking coffee [57])
Suppose a situation in which we reason about the effects of drinking a cup of coffee, given by the action theory $\mathcal{D}^{drink}_{coffee}$ such that:

$$\mathcal{S} = \emptyset,\ \mathcal{E}^{drink} = \left\{ \begin{array}{l} sugar \rightarrow [drink]happy, \\ salt \rightarrow [drink]\neg happy \end{array} \right\},$$

$$\mathcal{X}^{drink} = \mathcal{I}^{drink} = \emptyset$$

and the dependence relation is

$$\rightsquigarrow = \{\langle drink, happy\rangle, \langle drink, \neg happy\rangle\}$$

Observe that $\mathcal{D}^{drink}_{coffee}$ satisfies **PS**. Then, running Algorithm 7.2 on this action theory will give us $\mathcal{I}^{drink}_{imp} = \{(sugar \wedge salt) \rightarrow [drink]\bot\}$.

Remark 7.2 $\mathcal{I}^a \cup \mathcal{I}^a_{imp}$ is not always intuitive.

Whereas in Example 7.4 we have got an inexecutability that could be safely added to $\mathcal{I}^{tease}$, in Example 7.6 we got an inexecutability that is unintuitive (just the presence of sugar and salt in the coffee precludes drinking it). In that case, revision of other parts of the theory should be considered in order to make it intuitive. Anyway, the problem pointed out in the depicted scenario just illustrates that intuition is beyond syntax. The scope of this work relies on the syntactical level. Only the knowledge engineer can judge about how intuitive a formula is.

In the next chapter, we revisit our postulates in order to strengthen them to the case where more than one action is under concern, and thus get results that can be applied to whole action theories.

Chapter 8

where our method becomes a Swiss Army knife

Generalizing Modularity and Exploiting It

All generalizations are dangerous, even this one.

— Alexandre Dumas

In this chapter, we generalize Postulates **PC**, **PS** and **PI** to action theories as a whole, i.e., considering all actions of a domain, and prove some results that follow from that. We also investigate whether our set of postulates can be augmented in order to get a more refined notion of modularity. We close the chapter showing the benefits we get from domain descriptions that are modular in our sense.

8.1 Postulates for Multiple Action Theories

Go as far as you can see, and when you get there, you will see farther

— Anonymous

We have seen the importance satisfaction of Postulates **PC**, **PS** and **PI** may have in describing the action theory of a particular action a. However, in applications of real interest, more than one action is involved, and thus a natural question that could be raised is "can we have similar meta-theoretical results for multiple action theories?"

Given a dynamic domain, we define $\mathcal{E} = \bigcup_{a \in \mathfrak{Act}} \mathcal{E}^a$, $\mathcal{X} = \bigcup_{a \in \mathfrak{Act}} \mathcal{X}^a$, and $\mathcal{I} = \bigcup_{a \in \mathfrak{Act}} \mathcal{I}^a$. All these sets are finite, because $\mathfrak{Act}$ is finite and each of the $\mathcal{E}^a$, $\mathcal{X}^a$, $\mathcal{I}^a$ is finite. We here redefine action theories.

As I told earlier, I never repeat anything.

Definition 8.1 (Action theory)
An action theory *is a tuple* $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$*, where* $\mathcal{T} = \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I}$*.*

Given that, a generalization of Postulate **PC** for whole action theories is quite easy and has no need for justification:

PC* (**Logical consistency**): $\mathcal{D} \not\models \bot$

The whole action theory should be logically consistent.

Generalizing Postulate **PS** will give us the following:

PS* (**No implicit static laws**): if $\mathcal{D} \models \varphi$, then $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \models \varphi$

If a classical formula can be inferred from the whole action theory, then it should be inferable from the set of static laws alone. We have the following results:

Theorem 8.1
$\mathcal{D}$ *satisfies Postulate* **PS*** *if and only if* $\mathcal{D}^a$ *satisfies Postulate* **PS** *for all* $a \in \mathfrak{Act}$*.*

Proof:
($\Rightarrow$): Straightforward: Suppose that for some $a \in \mathfrak{Act}$ $\mathcal{D}^a$ does not satisfy **PS**. Then there is $\varphi \in \mathfrak{Fml}$ such that $\mathcal{D}^a \models \varphi$ and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \not\models \varphi$, i.e., $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi$ and $\mathcal{S} \not\models \varphi$. Of course $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\rightsquigarrow} \varphi$, by monotonicity, and then $\mathcal{D} \models \varphi$, but still $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \not\models \varphi$. Hence $\mathcal{D}$ does not satisfy **PS***.

($\Leftarrow$): Suppose $\mathcal{D}$ does not satisfy **PS***. Then there is $\varphi \in \mathfrak{Fml}$ such that $\mathcal{D} \models \varphi$ and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \not\models \varphi$, i.e., $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\rightsquigarrow} \varphi$ and $\mathcal{S} \not\models \varphi$. φ is equivalent to $\varphi_1 \wedge \ldots \wedge \varphi_n$, with $\varphi_1, \ldots, \varphi_n \in \mathfrak{Fml}$ and such that there is at least one φ_i such that $\mathcal{S} \not\models \varphi_i$ (otherwise $\mathcal{S} \models \varphi$). Because the logic is independently axiomatized, there must be some $a \in \mathfrak{Act}$ such that $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi_i$. From this and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \not\models \varphi_i$ it follows that $\mathcal{D}^a$ does not satisfy **PS**. ■

Corollary 8.1
$\mathcal{D}$ *satisfies Postulate* **PS*** *if and only if the big model for* $\mathcal{D}$ *is a model of* $\mathcal{D}$*.*

Proof:
The proof follows from Theorems 7.1 and 8.1. ■

Theorem 8.2
If $\mathcal{D}$ satisfies **PS***, *then $\mathcal{D}$ satisfies* **PC*** *if and only if $\mathcal{D}^a$ satisfies* **PC** *for all $a \in \mathfrak{Act}$.*

Proof:
Let $\mathcal{D}$ satisfy **PS***.

($\Rightarrow$): Suppose that $\mathcal{D}^a$ does not satisfy **PC**, for some $a \in \mathfrak{Act}$. Because $\mathcal{D}$ satisfies **PS***, $\mathcal{D}^a$ satisfies Postulate **PS** (Theorem 8.1), and then $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \models \bot$. From this it follows that $\mathcal{D} \models \bot$ (by monotonicity) and then $\mathcal{D}$ does not satisfy Postulate **PC***.

($\Leftarrow$): Suppose $\mathcal{D}$ does not satisfy **PC***. Then $\mathcal{D} \models \bot$. Because $\mathcal{D}$ satisfies Postulate **PS***, $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \models \bot$. Since $\mathfrak{Act} \neq \emptyset$, there is some $a \in \mathfrak{Act}$ such that $\mathcal{D}^a \models \bot$. ■

A more general form of Postulate **PI** can also be stated:

PI* (**No implicit inexecutability laws**):

$$\text{if } \mathcal{D} \models \varphi \rightarrow [a]\bot, \text{ then } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{I} \rangle \models \varphi \rightarrow [a]\bot$$

If an inexecutability law can be inferred from the whole action theory, then it should be inferable in PDL from the static and inexecutability laws alone.

Note that having that $\mathcal{D}^a$ satisfies **PI** for all $a \in \mathfrak{Act}$ is not enough to $\mathcal{D}$ satisfy **PI*** if there are implicit static laws. To witness, let $\mathcal{S} = \mathcal{E}^{a_1} = \emptyset$, $\mathcal{X}^{a_1} = \{\langle a_1 \rangle \top\}$, and $\mathcal{I}^{a_1} = \{\varphi \rightarrow [a_1]\bot\}$. Let also $\mathcal{E}^{a_2} = \mathcal{X}^{a_2} = \mathcal{I}^{a_2} = \emptyset$, and let $\rightsquigarrow = \emptyset$. Observe that both $\mathcal{D}^{a_1}$ and $\mathcal{D}^{a_2}$ satisfy **PI**, but $\mathcal{D} \models \varphi \rightarrow [a_2]\bot$ and $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{I} \rangle \not\models \varphi \rightarrow [a_2]\bot$.

Nevertheless, under **PS*** the result follows:

Theorem 8.3
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ satisfy Postulate **PS***. *$\mathcal{D}$ satisfies Postulate* **PI*** *if and only if $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$ satisfies Postulate* **PI** *for all $a \in \mathfrak{Act}$.*

Proof:
See Appendix D. ■

In the next section we make a step toward an attempt of amending our modularity criteria by investigating possible extensions of our set of postulates.

8.2 Can We Ask for More?

Can we augment our set of postulates to take into account other modules of action theories, or even other meta-theoretical issues in reasoning about actions? That is the topic we discuss in what follows.

Postulates about Action Effects

It seems to be in line with our postulates to require action theories not to allow for the deduction of new effect laws: if an effect law can be inferred from an action theory (and no inexecutability for the same action in the same context can be derived), then it should be inferable from the set of static and effect laws alone. This means that we should have:

PE (**No implicit effect laws**):

$$\text{if } \mathcal{D} \models \varphi \rightarrow [a]\psi \text{ and } \mathcal{D} \not\models \varphi \rightarrow [a]\bot, \text{ then } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \rangle \models \varphi \rightarrow [a]\psi$$

But consider the action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$ such that:

$$\mathcal{S} = \emptyset,\ \mathcal{E} = \left\{ \begin{array}{c} \mathit{loaded} \rightarrow [\mathit{shoot}]\neg \mathit{alive}, \\ (\neg \mathit{loaded} \wedge \mathit{alive}) \rightarrow [\mathit{shoot}]\mathit{alive} \end{array} \right\}$$

$$\mathcal{X} = \{\mathit{hasGun} \rightarrow \langle \mathit{shoot} \rangle \top\},\ \mathcal{I} = \{\neg \mathit{hasGun} \rightarrow [\mathit{shoot}]\bot\},$$

$$\rightsquigarrow = \{\langle \mathit{shoot}, \neg \mathit{alive} \rangle\}$$

Such a domain description satisfies Postulates **PS*** and **PI***, but does not satisfy **PE**. Indeed:

$$\mathcal{D} \models \neg \mathit{hasGun} \vee \mathit{loaded} \rightarrow [\mathit{shoot}]\neg \mathit{alive}$$

and

$$\mathcal{D} \not\models \neg \mathit{hasGun} \vee \mathit{loaded} \rightarrow [\mathit{shoot}]\bot,$$

but

$$\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \rangle \not\models \neg \mathit{hasGun} \vee \mathit{loaded} \rightarrow [\mathit{shoot}]\neg \mathit{alive}$$

So, Postulate **PE** would not help us to deliver the goods.

Another possibility of improving our modularity criteria could be:

P$\bot$ (**No unattainable effects**):

$$\text{if } \varphi \rightarrow [a]\psi \in \mathcal{E}, \text{ then } \mathcal{D} \not\models \varphi \rightarrow [a]\bot$$

This expresses that if we have explicitly stated an effect law for a in some context, then there should be no inexecutability law for the same action in the same context.

It is straightforward to design an algorithm which checks whether this postulate is satisfied. We do not investigate this further here, but just observe that the slightly stronger version below leads to unintuitive consequences:

P⊥′ (No unattainable effects – strong version):

$$\text{if } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \rangle \models \varphi \rightarrow [a]\psi, \text{ then } \mathcal{D} \not\models \varphi \rightarrow [a]\bot$$

Indeed, for the above action theory we have

$$\mathcal{E} \models_{\rightsquigarrow} (\neg \mathit{hasGun} \wedge \mathit{loaded}) \rightarrow [\mathit{shoot}]\neg \mathit{alive},$$

but

$$\mathcal{D} \models (\neg \mathit{hasGun} \wedge \mathit{loaded}) \rightarrow [\mathit{shoot}]\bot.$$

This is certainly too strong. Our example also illustrates that it is sometimes natural to have some "redundancies" or "overlaps" between $\mathcal{E}$ and $\mathcal{I}$. Indeed, as we have pointed out, inexecutability laws are a particular kind of effect laws, and the distinction here made is conventional. The decision of considering them as strictly different entities or not depends mainly on the context. At a representational level, in the most part of this work, we preferred to keep them separated, while in Algorithm 7.1 we have mixed them together in order to compute all the consequences of an action (cf. Chapters 4 and 9).

In what follows we address the problem of completing the set of executability laws of an action theory.

Maximizing Executability

As we have seen, implicit static laws only show up when there are executability laws. So, a question that naturally raises is "which executability laws can be consistently added to a given action theory?"

A hypothesis usually made in the literature is that of maximization of executabilities: in the absence of a proof that an action is inexecutable in a given context, assume its executability for that context. Such a hypothesis is formally captured by the following postulate:

PX$^+$ (Maximal executability laws):

$$\text{if } \mathcal{D} \not\models \varphi \rightarrow [a]\bot, \text{ then } \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{X} \rangle \models \varphi \rightarrow \langle a \rangle \top$$

Such a postulate expresses that if in context φ no inexecutability for a can be inferred, then the respective executability should follow in PDL from the executability and static laws.

Postulate **PX$^+$** generally holds in nonmonotonic frameworks, and can be enforced in monotonic approaches such as ours by maximizing $\mathcal{X}$. We nevertheless would like to point out that maximizing executability is not always intuitive. To witness, suppose we know that if we have the ignition key, the tank is full, ..., and the battery tension is beyond 10V, then the car (necessarily) will start. Suppose we also know that if the tension is below 8V, then the car will not start. What should we conclude in situations where we know that the tension is 9V? Maximizing executabilities makes us infer that it will start, but such reasoning is not what we want if we would like to be sure that all possible executions lead to the goal (cf. Section 1.1).

We do not investigate this further here, and in the rest of the chapter we emphasize the main results that we obtain when our modularity principle is satisfied.

8.3 The Role of Modularity in Reasoning

We start by generalizing the definition of modularity for multiple action theories.

Oh, déjà vu!

— Neo, in Matrix

Definition 8.2 (Modularity)
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{T} \rangle$ be an action theory such that $\mathcal{T} = \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I}$. $\mathcal{D}$ is modular *if and only if*

1. $\mathcal{D} \models \varphi$ *implies* $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \models \varphi$
2. $\mathcal{D} \models \varphi \rightarrow \langle a \rangle \top$ *implies* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{X} \rangle \models \varphi \rightarrow \langle a \rangle \top$
3. $\mathcal{D} \models \varphi \rightarrow [a]\bot$ *implies* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{I} \rangle \models \varphi \rightarrow [a]\bot$
4. $\mathcal{D} \models \varphi \rightarrow [a]\psi$ *implies* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{I} \rangle \models \varphi \rightarrow [a]\psi$

In what follows, we see how modularity can be reduced to our base postulates.

Theorem 8.4
If $\mathcal{D}$ satisfies Postulate **PS***, *then $\mathcal{D} \models \bot$ if and only if $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \models \bot$.*

This theorem says that if there are no implicit static laws, then consistency of an action theory can be checked by just checking consistency of $\mathcal{S}$. An immediate consequence is that consistency of a new learned information φ w.r.t. the whole description can be checked by just checking consistency of $\mathcal{S} \cup \{\varphi\}$.

Progress isn't made by early risers. It's made by lazy men trying to find easier ways to do something.

— Robert Heinlein

Theorem 8.5
If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ satisfies Postulate **PS***, *then $\mathcal{D} \models \varphi \rightarrow [a]\psi$ if and only if $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle \models \varphi \rightarrow [a]\psi$.*

Proof:
See Appendix D. ■

This means that under **PS*** we have modularity inside $\mathcal{E}$, too: when deducing the effects of a, we need not consider the action laws for other actions.

Versions of Theorem 8.5 for executability and inexecutability can be stated as well:

Theorem 8.6
If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ satisfies Postulate **PS***, *then $\mathcal{D} \models \varphi \rightarrow \langle a \rangle \top$ if and only if $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{X}^a \rangle \models \varphi \rightarrow \langle a \rangle \top$.*

Proof:
See Appendix D. ■

Corollary 8.2
Postulate **PX** *is a consequence of* **PS**.

Proof:
Straightforward. ■

Hence, Item 2 in Definition 8.2 is subsumed by Item 1. With this and Theorem 8.5 above we get that modularity of action theories in reasoning about actions amounts to having neither implicit static laws nor implicit inexecutability laws in the theory.

Theorem 8.7
If $\mathcal{D}$ is modular, then $\mathcal{D} \models \varphi \rightarrow [a]\bot$ if and only if $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle \models \varphi \rightarrow [a]\bot$.

Proof:
($\Rightarrow$): If $\mathcal{D} \models \varphi \rightarrow [a]\bot$, then $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\rightsquigarrow} \varphi \rightarrow [a]\bot$, and from **PS*** and Theorem 8.5 we have $\mathcal{S}, \mathcal{E}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi \rightarrow [a]\bot$. From this and **PI*** we get $\mathcal{S}, \mathcal{I}^a \models_{\mathsf{PDL}} \varphi \rightarrow [a]\bot$, from what the result follows.

($\Leftarrow$): Suppose $\mathcal{D} \not\models \varphi \rightarrow [a]\bot$, i.e., $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \rightarrow [a]\bot$. Then there is a $\rightsquigarrow$-model $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$ and $\not\models^{\mathscr{M}} \varphi \rightarrow [a]\bot$. Then, given a, we have $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{X}^a \wedge \mathcal{I}^a$, and then $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{I}^a$. Moreover, by definition, $\mathscr{M}$ is a PDL-model. Hence $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} \varphi \rightarrow [a]\bot$, and then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle \not\models \varphi \rightarrow [a]\bot$. ■

In Theorems 8.6 and 8.7, modularity guarantees that no dependence is needed to derive, respectively, executabilities and inexecutabilities.

Remark 8.1 There exist action theories $\mathcal{D}$ not satisfying Postulate **PS*** such that both $\mathcal{D} \models \varphi \rightarrow [a]\psi$ and $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle \not\models \varphi \rightarrow [a]\psi$.

As an example, for $\mathcal{D}_{wts}$ such that

$$\mathcal{S} = \{\mathit{walking} \rightarrow \mathit{alive}\},\ \mathcal{E} = \left\{ \begin{array}{c} [\mathit{tease}]\mathit{walking}, \\ \mathit{loaded} \rightarrow [\mathit{shoot}]\neg\mathit{alive}, \end{array} \right\},$$

$$\mathcal{X} = \{\langle \mathit{tease} \rangle \top\},\ \mathcal{I} = \{\neg\mathit{alive} \rightarrow [\mathit{tease}]\bot\}$$

and

$$\rightsquigarrow = \left\{ \begin{array}{c} \langle \mathit{shoot}, \neg\mathit{loaded} \rangle, \langle \mathit{shoot}, \neg\mathit{alive} \rangle, \\ \langle \mathit{shoot}, \neg\mathit{walking} \rangle, \langle \mathit{tease}, \mathit{walking} \rangle \end{array} \right\}$$

we have that

$$\mathcal{D} \models \neg\mathit{alive} \rightarrow [\mathit{shoot}]\mathit{alive},$$

but

$$\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{\mathit{shoot}} \cup \mathcal{I}^{\mathit{shoot}} \rangle \not\models \neg\mathit{alive} \rightarrow [\mathit{shoot}]\mathit{alive}.$$

Let $\mathcal{E}^{a_1,\ldots,a_n} = \bigcup_{1 \leq i \leq n} \mathcal{E}^{a_i}$, $\mathcal{X}^{a_1,\ldots,a_n} = \bigcup_{1 \leq i \leq n} \mathcal{X}^{a_i}$, and $\mathcal{I}^{a_1,\ldots,a_n} = \bigcup_{1 \leq i \leq n} \mathcal{I}^{a_i}$. Under Postulate **PS***, deduction of an effect of a sequence of actions $a_1; \ldots; a_n$ (prediction) needs neither the effect and inexecutability laws for actions other than $a_1, \ldots, a_n$, nor the executability laws of the domain:

Theorem 8.8
If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ *satisfies Postulate* **PS***, *then* $\mathcal{D} \models \varphi \rightarrow [a_1; \ldots; a_n]\psi$ *if and only if* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n} \rangle \models \varphi \rightarrow [a_1; \ldots; a_n]\psi$.

Proof:
See Appendix D ■

The same result holds for testing inexecutability of a sequence of actions:

Corollary 8.3
If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ *satisfies Postulate* **PS***, *then* $\mathcal{D} \models \varphi \rightarrow [a_1; \ldots; a_n]\bot$ *if and only if* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n} \rangle \models \varphi \rightarrow [a_1; \ldots; a_n]\bot$.

Proof:
Straightforward, as a special case of Theorem 8.8. ■

The next theorem shows that our notion of modularity is also fruitful in plan validation:

Theorem 8.9
If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ *satisfies Postulate* **PS***, *then* $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_n \rangle \psi$ *if and only if* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{X}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n} \rangle \models \varphi \rightarrow \langle a_1; \ldots; a_n \rangle \psi$.

Proof:
See Appendix D. ■

And as a consequence, we also optimize testing executability of a plan:

Corollary 8.4
If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ *satisfies Postulate* **PS***, *then* $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_n \rangle \top$ *if and only if* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{X}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n} \rangle \models \varphi \rightarrow \langle a_1; \ldots; a_n \rangle \top$.

Proof:
Straightforward, as a special case of Theorem 8.9. ■

Theorems 8.8 and 8.9 together with Corollaries 8.3 and 8.4 suggest that we can simulate modularization by sub-domains [77]: If $\langle \{a_1, \ldots, a_n\}, \mathfrak{Prop}' \rangle$ is a sub-domain for some $\mathfrak{Prop}' \subseteq \mathfrak{Prop}$, then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{X}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n} \rangle$ corresponds to the module for $\langle \{a_1, \ldots, a_n\}, \mathfrak{Prop}' \rangle$ in Lifschitz and Ren's sense (cf. Section 10.2).

In the following chapter, we investigate the role modularity plays when the domain description has to be changed.

Chapter 9

where we cut, crop and paste with our Swiss Army knife

Towards Action Theory Change

Education consists mainly in what we have unlearned.

— Mark Twain

We here address the problem of *changing* action theories and define a general method based on contraction of formulas. We present the semantics of our theory change and define syntactical operators for contracting a domain description. We establish soundness and completeness of the operators w.r.t. the semantics for descriptions that satisfy our principle of modularity. We also investigate an example of changing non-modular domain descriptions.

9.1 Motivation

Suppose a situation where an agent has always believed that if the light switch is up, then there is light in the room. Suppose now that someday she observes that even if the switch is in the upper position, the light is off. In such a case, the agent must change her beliefs about the relation between the propositions "the switch is up" and "the light is on". This example is an instance of the problem of changing propositional belief bases and is largely addressed in the literature about belief revision [37] and belief update [67].

Next, let our agent believe that whenever the switch is down, after toggling it, there is light in the room. This means that if the light is off, in every state of the world that follows the execution of toggling the switch, the room is lit up. Then, during a blackout, the agent toggles the switch and surprisingly the room is still dark.

Imagine now that the agent never worried about the relation between toggling

the switch and the material it is made of, in the sense that she ever believed that just toggling the switch does not break it. Nevertheless, in a stressful day, she toggles the switch and then observes that she had broken it.

Completing the wayside cross our agent experiments in discovering the world's behavior, suppose she ever believed it is always possible to toggle the switch, provided some conditions like being close enough to it, having a free hand, that the switch is not broken, etc, are satisfied. Then, in a beautiful April fool's day, she discovers that someone has glued the switch and consequently it is no longer possible to toggle it.

The last three examples illustrate situations where changing the beliefs about the behavior of the action of toggling the switch is mandatory. In the first one, toggling the switch, once believed to be deterministic, has now to be seen as nondeterministic, or, alternatively, to have a different outcome in a specific context (e.g. if the power station is overloaded). In the second example, toggling the switch is known to have side-effects (ramifications) one was not aware of. In the last example, the executability of the action under concern is questioned in the light of new information showing a context that was not known to preclude its execution.

Such cases of theory change are very important when one deals with logical descriptions of dynamic domains: it may always happen that one discovers that an action actually has a behavior that is different from that one has always believed it had.

Up to now, theory change has been studied mainly for knowledge bases in classical logics, both in terms of revision and update. Only in a few recent works it has been considered in modal logics, viz. in epistemic logic [48], and in action languages [31]. Recently, some works [106, 62] have investigated revision of beliefs about facts of the world. In our examples, this would concern e.g. the current status of the switch: the agent believes it is up, but is wrong about this and might subsequently be forced to revise her beliefs about the current state of affairs. Such revision operations do not modify the agent's beliefs about action laws. In opposition to that, here we are interested exactly in such modifications. Our aim in this chapter is to make a step toward that issue and propose a framework that deals with contraction of action theories.

9.2 Models of Contraction

When a domain description has to be changed, the basic operation is that of *contraction*. (In belief-base update [116, 67] it has also been called *erasure*.) In this section, we define its semantics.

For the sake of presentation, as in Chapter 4 we here consider inexecutability laws as special cases of effect laws (those whose effect is $\bot$). It can be seen that doing things this way does no harm to the theoretical results we have obtained so far.

In general, we might contract by any formula Φ. Here we focus on contraction by one of the three kinds of laws. We therefore suppose that Φ is either φ, where φ is classical, or $\varphi \to [a]\psi$, or $\varphi \to \langle a\rangle\top$. The contraction of a model $\mathscr{M} = \langle W, R\rangle$ by Φ results in a set of models each of which is a minimal modification of $\mathscr{M}$ that is no longer a model of Φ.

For the case of contracting static laws, we resort to existing approaches in order to change the set of static laws. In the following, we consider any belief change operator such as Forbus' update method [36], or the possible models approach [116, 117], or WSS [53] or MPMA [27].

Contraction by φ corresponds to adding new possible worlds to W. Let $\ominus$ be a contraction operator for classical logic.

Definition 9.1 (Semantics of classical contraction)
Let $\mathscr{M} = \langle W, R\rangle$ be a PDL*-model and φ a classical formula. The set of models resulting from contracting $\mathscr{M}$ by φ is the singleton $\mathscr{M}^-_\varphi = \{\langle W', R\rangle\}$ such that $W' = W \ominus \mathit{valuations}(\varphi)$.*

For example, consider the model $\mathscr{M}$ in Figure 9.1 (note that $\models^{\mathscr{M}} p_1 \to p_2$) and suppose that we want to contract $\mathscr{M}$ by the static law $p_1 \to p_2$. The result is depicted by $\mathscr{M}'$ in Figure 9.1, with $\mathscr{M}' \in \mathscr{M}^-_{p_1 \to p_2}$.

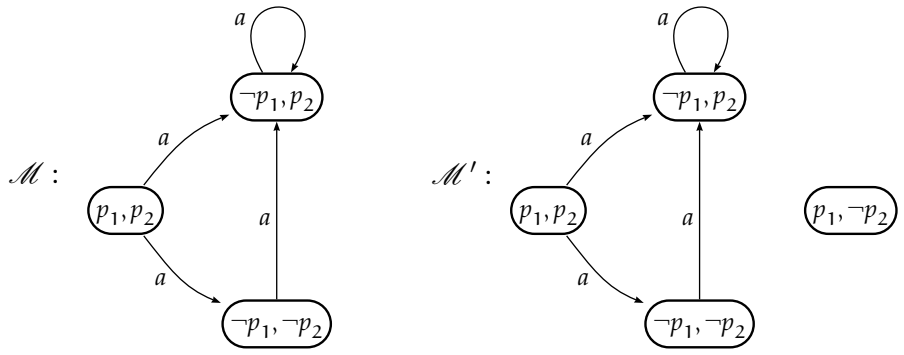

Figure 9.1: Contraction of a model by a static law.

Observe that the accessibility relation R should, a priori, change as well. Figure 9.2 shows two models resulting from contracting $\mathscr{M}$ in Figure 9.1 by the static law $p_1 \to p_2$

in which R has been changed so that we have arrows leaving the world just added.

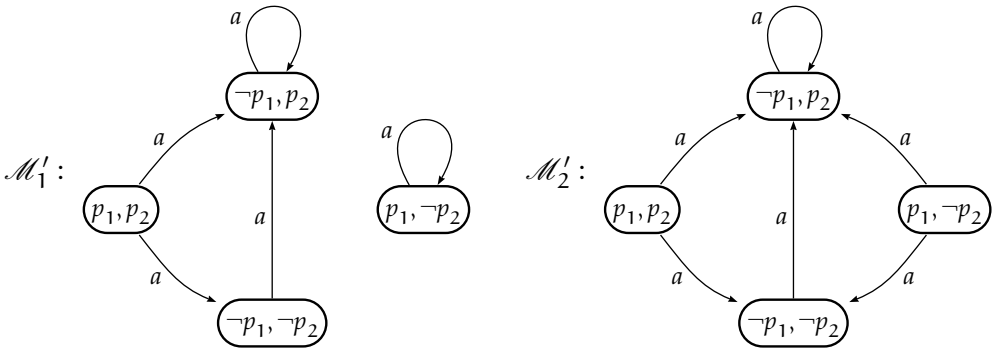

Figure 9.2: Contraction of a static law: adding leaving arrows to the new world.

The reason for changing R is that otherwise contracting a classical formula may conflict with $\mathcal{X}$. For instance, if $\neg\varphi \rightarrow \langle a\rangle\top \in \mathcal{X}$ and we contract by φ, the result may make $\mathcal{X}$ untrue. However, given the amount of information we have at hand, we think that whatever we do with R (adding or removing edges), we will always be able to find a counter-example to the intuitiveness of the operation, since it is domain dependent. For instance, adding edges for a deterministic action may render it non-deterministic. Deciding on what changes to carry out on R when contracting static laws depends on the user's intuition, and unfortunately this information cannot be generalized and established once for all. We here opt for a priori doing nothing with R and postponing correction of executability laws.

Action theories being defined in terms of effect and executability laws, changing an action theory will mainly involve changing one of these two sets of laws. Let us consider now both these cases.

Suppose the knowledge engineer acquires new information regarding the effect of action a. Then it means that the law under consideration is probably too strong, i.e., the expected effect may not occur and thus the law has to be weakened. Consider e.g. $\neg up \rightarrow [toggle]light$, and suppose it has to be weakened to the more specific $(\neg up \wedge \neg blackout) \rightarrow [toggle]light$.[1] In order to carry out such a weakening, first the designer has to contract the set of effect laws and second to expand the resulting set with the weakened law.

[1]The other possibility of weakening the law, i.e., replacing it by $\neg up \rightarrow [toggle](light \vee \neg light)$ looks silly. We were not able to find examples where changing the consequent could give a more intuitive result. In this sense, we prefer to always weaken a given law by strengthening its antecedent.

Contraction by $\varphi \to [a]\psi$ amounts to adding some "counter-example" arrows from φ-worlds to $\neg\psi$-worlds.

Definition 9.2 (Semantics of effect contraction)
Let $\mathscr{M} = \langle W, R\rangle$ be a PDL*-model and $\varphi \to [a]\psi$ an effect law. The models resulting from contracting $\mathscr{M}$ by $\varphi \to [a]\psi$ is $\mathscr{M}^{-}_{\varphi\to[a]\psi} = \{\langle W, R \cup R'_a\rangle \ : \ R'_a \subseteq \{(w,w') \ : \models^{\mathscr{M}}_{w} \varphi\}\}$.*

Figure 9.3 depicts the three resulting models of contracting $\neg p_2 \to [a]p_2$ in the model $\mathscr{M}$ of Figure 9.1.

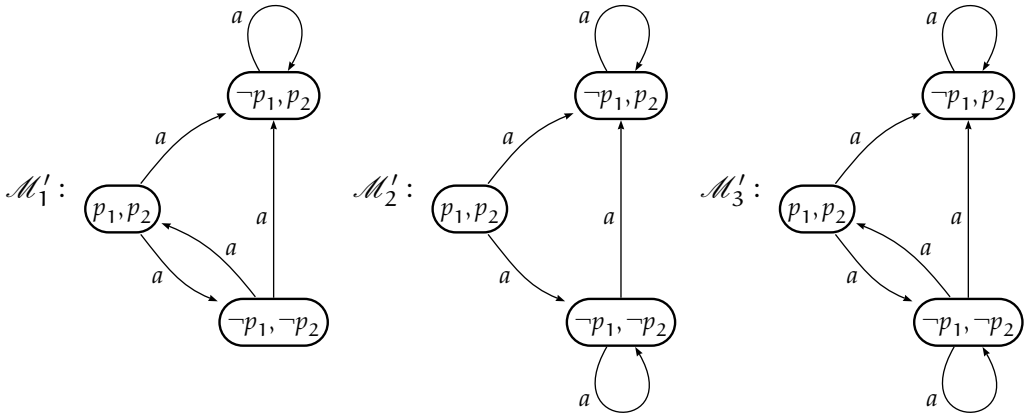

Figure 9.3: Contraction of model $\mathscr{M}$ in Figure 9.1 by an effect law.

Suppose now the knowledge engineer learns new information about the executability of a. This usually occurs when there are executability laws that are too strong, i.e., the condition in the theory guaranteeing the executability of a is too weak and has to be made more restrictive. Let e.g. $\langle toggle\rangle\top$ be the law to be contracted, and suppose it has to be weakened to the more specific $\neg broken \to \langle toggle\rangle\top$. To implement such a weakening, the designer has to first contract the set of executability laws and then to expand the resulting set with the weakened law.

Contraction by $\varphi \to \langle a\rangle\top$ corresponds to removing some arrows leaving worlds where φ holds. Removing such arrows has as consequence that a is no longer always executable in context φ.

Definition 9.3 (Semantics of executability contraction)
Let $\mathscr{M} = \langle W, R\rangle$ be a PDL*-model and $\varphi \to \langle a\rangle\top$ an executability law. The set of models resulting from contracting $\mathscr{M}$ by $\varphi \to \langle a\rangle\top$ is $\mathscr{M}^{-}_{\varphi\to\langle a\rangle\top} = \{\langle W, R \setminus R'_a\rangle \ : \ R'_a \subseteq \{(w,w') \ : \ wR_aw' \text{ and } \models^{\mathscr{M}}_{w} \varphi\}\}$.*

Figure 9.4 illustrates contraction of model $\mathscr{M}$ in Figure 9.1 by the executability $p_1 \to \langle a \rangle \top$. (Observe that, in this example, $p_1 \to [a]\bot$ has not become true in $\mathscr{M}^-_{p_1 \to \langle a \rangle \top}$, but it is not hard to imagine examples where an inexecutability becomes valid when contracting an executability law.)

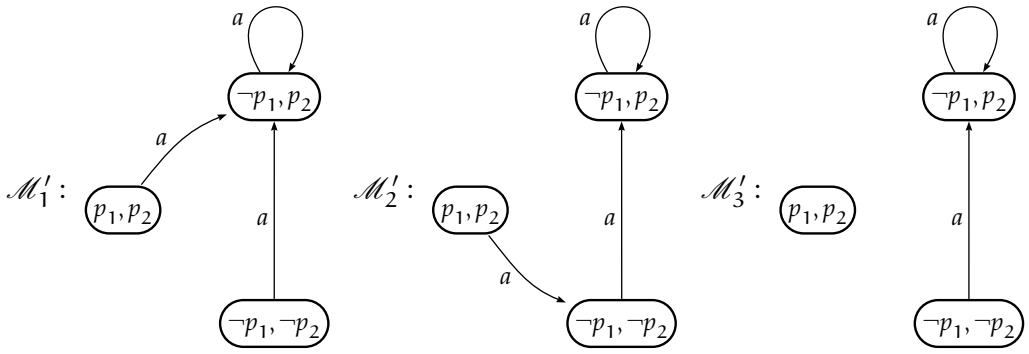

Figure 9.4: Contraction of model $\mathscr{M}$ in Figure 9.1 by an executability law.

In the next section, we make a step toward syntactical operators that reflect the semantic foundations for contraction.

9.3 Contracting an Action Theory

Having established the semantics of action theory contraction, we can turn to its syntactical counterpart.

Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \rangle$ be an action theory and Φ a PDL-formula. By $\mathcal{D}^-_\Phi$ we denote the action theory resulting from the contraction of $\mathcal{D}$ by Φ.

Contracting a theory by a static law φ amounts to using any existing contraction operator for classical logic. Let $\ominus$ be such an operator. Moreover, we also need to guarantee that φ will not continue to follow from $\mathcal{E}$, $\mathcal{X}$ and $\rightsquigarrow$, i.e., in the case φ is an implicit static law (cf. Sections 7.3 and 9.4). We define contraction of a domain description by a static law as follows:

Definition 9.4 (Contraction of a static law)
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \rangle$. $\mathcal{D}^-_\varphi = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S}^- \cup \mathcal{E} \cup \mathcal{X}^- \rangle$, *where* $\mathcal{S}^- = \mathcal{S} \ominus \varphi$ *and* $\mathcal{X}^- = (\mathcal{X} \setminus \mathcal{X}^a) \cup \{(\varphi_i \wedge \varphi) \to \langle a \rangle \top \,:\, \varphi_i \to \langle a \rangle \top \in \mathcal{X}^a\}$.

For example, contracting the law $up \to light$ in our running scenario, besides changing $\mathcal{S}$, would give us $\mathcal{X}^- = \{(\neg up \vee light) \to \langle toggle \rangle \top\}$, so that the old exe-

cutabilities are still satisfied in the new possible state $\{up, \neg light\}$ that is intended to be added at the semantical level.

To contract a theory by $\varphi \to [a]\psi$, for every effect law in $\mathcal{D}$, we must ensure that a still has effect ψ whenever φ does not hold, and change $\rightsquigarrow$ so that a may have $\neg\psi$ as outcome. This is enough to guarantee that the law has been contracted. The operator below formalizes this:

Definition 9.5 (Contraction of an effect law)
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \rangle$. $\mathcal{D}^{-}_{\varphi \to [a]\psi} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow'}, \mathcal{S} \cup \mathcal{E}^{-} \cup \mathcal{X} \rangle$, *where* $\mathcal{E}^{-} = (\mathcal{E} \setminus \mathcal{E}^{a}) \cup \{(\varphi_i \wedge \neg\varphi) \to [a]\psi_i : \varphi_i \to [a]\psi_i \in \mathcal{E}^{a}\}$, *and* $\rightsquigarrow' = \rightsquigarrow \cup (\{a\} \times \mathfrak{Lit})$.

If $\mathcal{D}_{light}$ denotes our running example such that

$$\mathcal{S} = \{up \to light\},\ \mathcal{E} = \left\{ \begin{array}{c} \neg up \to [toggle]up, \\ up \to [toggle]\neg up \end{array} \right\},$$

$$\mathcal{X} = \{\langle toggle \rangle \top\},\ \rightsquigarrow = \left\{ \begin{array}{c} \langle toggle, light \rangle, \langle toggle, \neg light \rangle, \\ \langle toggle, up \rangle, \langle toggle, \neg up \rangle \end{array} \right\}$$

then contracting the law $blackout \to [toggle]light$ from $\mathcal{D}_{light}$ would give us

$$\mathcal{E}^{-} = \left\{ \begin{array}{c} (\neg up \wedge \neg blackout) \to [toggle]up, \\ (up \wedge \neg blackout) \to [toggle]\neg up \end{array} \right\},$$

$$\rightsquigarrow' = \left\{ \begin{array}{c} \langle toggle, light \rangle, \langle toggle, \neg light \rangle, \\ \langle toggle, up \rangle, \langle toggle, \neg up \rangle, \\ \langle toggle, blackout \rangle, \langle toggle, \neg blackout \rangle \end{array} \right\}$$

Finally, we consider the case of contracting an action theory by an executability law $\varphi \to \langle a \rangle \top$. For every executability in $\mathcal{D}$, we ensure that action a is executable only in contexts where $\neg\varphi$ is the case. The following operator does the job.

Definition 9.6 (Contraction of an executability law)
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \rangle$. $\mathcal{D}^{-}_{\varphi \to \langle a \rangle \top} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X}^{-} \rangle$, *where* $\mathcal{X}^{-} = (\mathcal{X} \setminus \mathcal{X}^{a}) \cup \{(\varphi_i \wedge \neg\varphi) \to \langle a \rangle \top : \varphi_i \to \langle a \rangle \top \in \mathcal{X}^{a}\}$.

For instance, contracting the executability $glued \to \langle toggle \rangle \top$ from $\mathcal{D}_{light}$ would give us $\mathcal{X}^{-} = \{\neg glued \to \langle toggle \rangle \top\}$.

Now we establish that our operators are correct w.r.t. the semantics. Our first

theorem establishes that the semantical contraction of the models of $\mathcal{D}$ by Φ produces models of the contracted theory $\mathcal{D}^{-}_{\Phi}$.

Theorem 9.1
Let Φ be a formula that has the form of one of the three laws. For all models $\mathscr{M}'$, if $\mathscr{M}' \in \mathscr{M}^{-}_{\Phi}$ for some $\mathscr{M} = \langle W, R\rangle$ such that $\models^{\mathscr{M}} \mathcal{D}$, then $\models^{\mathscr{M}'} \mathcal{D}^{-}_{\Phi}$.

Proof:
See Appendix E. ■

It remains to prove that the other way round, the models of $\mathcal{D}^{-}_{\Phi}$ result from the semantical contraction of models of $\mathcal{D}$ by Φ. This does not hold in general, as shown by the following example: suppose there is only one atom p and one action a, and consider the action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X}\rangle$ such that $\mathcal{S} = \emptyset$, $\mathcal{E} = \{p \rightarrow [a]\bot\}$, $\mathcal{X} = \{\langle a\rangle\top\}$, and $\rightsquigarrow = \emptyset$. The only model of that action theory is $\mathscr{M} = \langle\{\{\neg p\}\}, \{(\{\neg p\}, \{\neg p\})\}\rangle$ in Figure 9.5. By definition, $\mathscr{M}^{-}_{p\rightarrow\langle a\rangle\top} = \{\mathscr{M}\}$. On the other hand, $\mathcal{D}^{-}_{p\rightarrow\langle a\rangle\top}$ is such that $\mathcal{S} = \rightsquigarrow = \emptyset$, $\mathcal{E} = \{p \rightarrow [a]\bot\}$, and $\mathcal{X} = \{\neg p \rightarrow \langle a\rangle\top\}$. The contracted theory has *two* models: $\mathscr{M}$ and $\mathscr{M}' = \langle\{\{p\}, \{\neg p\}\}, (\{\neg p\}, \{\neg p\})\rangle$ in Figure 9.5. While $\neg p$ is valid in the contraction of the models of $\mathcal{D}$, it is not valid in the models of $\mathcal{D}^{-}_{p\rightarrow\langle a\rangle\top}$.

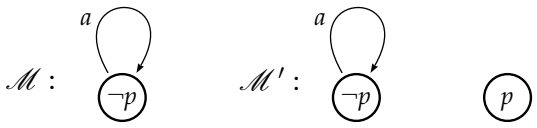

Figure 9.5: Incompleteness of contraction.

Fortunately, we can establish a result for those action theories that are modular. The proof requires three lemmas. The first one says that for a modular theory we can restrict our attention to its big models.

Lemma 9.1
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X}\rangle$ be modular. Then $\mathcal{D} \models \Phi$ if and only if $\models^{\mathscr{M}} \Phi$ for every model $\mathscr{M} = \langle W, R\rangle$ of $\mathcal{D}$ such that $W = \mathit{valuations}(\mathcal{S})$.

Proof:
($\Rightarrow$): Because $\mathcal{D}$ is modular, $\mathcal{D}$ satisfies Postulate **PS***. By Corollary 8.2, for every $\mathscr{M} = \langle W, R\rangle$ such that $W = \mathit{valuations}(\mathcal{S})$, $\models^{\mathscr{M}} \mathcal{D}$. From the hypothesis $\mathcal{D} \models \Phi$, it follows $\models^{\mathscr{M}} \Phi$.

($\Leftarrow$): Suppose $\mathcal{D} \not\models \Phi$. Then there is a model $\mathscr{M} = \langle W, R\rangle$ such that $\models^{\mathscr{M}} \mathcal{D}$ and $\not\models^{\mathscr{M}} \Phi$. We can augment $\mathscr{M}$ to a big model $\mathscr{M}' = \langle \mathit{valuations}(\mathcal{S}), R\rangle$. Because $\mathcal{D}$ is modular, by Corollary 8.2, it follows $\models^{\mathscr{M}'} \mathcal{D}$. Clearly $\not\models^{\mathscr{M}'} \Phi$. ∎

Note that the lemma does not hold for non-modular theories (because the set $\{\langle W, R\rangle \ : \ W = \mathit{valuations}(\mathcal{S})\}$ is empty then).

The second lemma says that modularity is preserved under contraction.

Lemma 9.2
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X}\rangle$ be modular, and let Φ be a formula of the form of one of the three laws. Then $\mathcal{D}^{-}_{\Phi}$ is modular.

Proof:
See Appendix E. ∎

The third one establishes the required link between the contraction operators and contraction of big models.

Lemma 9.3
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X}\rangle$ be modular, let Φ be a formula of the form of one of the three laws, and $\mathcal{D}^{-}_{\Phi} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, , \mathcal{S}' \cup \mathcal{E}' \cup \mathcal{X}'\rangle$. If $\mathscr{M}' = \langle \mathit{valuations}(\mathcal{S}'), R'\rangle$ is a model of $\mathcal{D}^{-}_{\Phi}$, then there is a model $\mathscr{M}$ of $\mathcal{D}$ such that $\mathscr{M}' \in \mathscr{M}^{-}_{\Phi}$.

Proof:
Let $\mathscr{M}' = \langle \mathit{valuations}(\mathcal{S}'), R'\rangle$ be such that $\models^{\mathscr{M}'} \mathcal{D}^{-}_{\Phi}$. We analyze each case.

Let Φ be φ, for some propositional $\varphi \in \mathfrak{Fml}$. Because $\mathcal{D}$ is modular, Lemma 9.1 gives us that there is a model $\mathscr{M} = \langle \mathit{valuations}(\mathcal{S}), R\rangle$ such that $\models^{\mathscr{M}} \mathcal{D}$. Clearly, $\mathscr{M}' \in \mathscr{M}^{-}_{\varphi}$, from soundness of $\ominus$.

Suppose now Φ has the form $\varphi \rightarrow [a]\psi$, for $\varphi, \psi \in \mathfrak{Fml}$. $\mathcal{D}$ being modular, Lemma 9.1 gives us that $\mathscr{M} = \langle \mathit{valuations}(\mathcal{S}), R\rangle$ is such that $\models^{\mathscr{M}} \mathcal{D}$. Because, when contracting effect laws, $\mathcal{S}' = \mathcal{S}$, it suffices to choose R and R''_a such that $R' = R \cup R''_a$, for some $R''_a \subseteq \{(w, w') \ : \models^{\mathscr{M}}_{w} \varphi\}$, and then $\mathscr{M}' \in \mathscr{M}^{-}_{\varphi \rightarrow [a]\psi}$.

Now let Φ have the form $\varphi \rightarrow \langle a\rangle \top$, for some $\varphi \in \mathfrak{Fml}$. From $\mathcal{D}$ modular and Lemma 9.1, there is $\mathscr{M} = \langle \mathit{valuations}(\mathcal{S}), R\rangle$ such that $\models^{\mathscr{M}} \mathcal{D}$. When contracting executabilities, $\mathcal{S}' = \mathcal{S}$, hence taking the right R and R''_a such that $R' = R \setminus R''_a$, for some $R_a{}'' \subseteq \{(w, w') \ : \ wR_aw' \text{ and } \models^{\mathscr{M}}_{w} \varphi\}$, we get $\mathscr{M}' \in \mathscr{M}^{-}_{\varphi \rightarrow \langle a\rangle \top}$. ∎

Putting the three above lemmas together we get:

Theorem 9.2
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \rangle$ *be modular, let* Φ *be a formula of the form of one of the three laws, and* $\mathcal{D}^-_\Phi = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, , \mathcal{S}' \cup \mathcal{E}' \cup \mathcal{X}' \rangle$. *For all models* $\mathscr{M}'$, *if* $\models^{\mathscr{M}'} \mathcal{D}^-_\Phi$, *then* $\mathscr{M}' \in \mathscr{M}^-_\Phi$, *for some* $\mathscr{M} = \langle W, R \rangle$ *such that* $\models^{\mathscr{M}} \mathcal{D}$.

Proof:
From the hypothesis that $\mathcal{D}$ is modular and Lemma 9.2, $\mathcal{D}^-_\Phi$ is modular. Then, $\mathscr{M}' = \langle \mathit{valuations}(\mathcal{S}'), R' \rangle$ is such that $\models^{\mathscr{M}'} \mathcal{D}^-_\Phi$, by Lemma 9.1. From this and Lemma 9.3, the result follows. ■

Our two theorems together establish correctness of the operators:

Corollary 9.1
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \rangle$ *be modular, let* Φ *be a formula of the form of one of the three laws, and* $\mathcal{D}^-_\Phi = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, , \mathcal{S}' \cup \mathcal{E}' \cup \mathcal{X}' \rangle$. *Then* $\mathcal{D}^-_\Phi \models \Psi$ *if and only if for every model* $\mathscr{M}'$ *such that* $\mathscr{M}' \in \mathscr{M}^-_\Phi$ *for some* $\mathscr{M}$ *such that* $\models^{\mathscr{M}} \mathcal{D}$, $\models^{\mathscr{M}'} \Psi$.

Proof:
($\Rightarrow$): Let $\mathscr{M}'$ be such that $\models^{\mathscr{M}'} \mathcal{D}^-_\Phi$. By Theorem 9.2, $\mathscr{M}' \in \mathscr{M}^-_\Phi$ for some $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{D}$. From the hypothesis $\mathcal{D}^-_\Phi \models \Psi$, we have $\models^{\mathscr{M}'} \Psi$.

($\Leftarrow$): Suppose that $\mathcal{D}^-_\Phi \not\models \Psi$. Then there is a model $\mathscr{M} = \langle W, R \rangle$ such that $\models^{\mathscr{M}} \mathcal{D}^-_\Phi$ and $\not\models^{\mathscr{M}} \Psi$. Because $\mathcal{D}$ is modular, by Lemma 9.2, $\mathcal{D}^-_\Phi$ is modular, too. By Lemma 9.1, $\mathscr{M}$ can be augmented to a big model $\mathscr{M}' = \langle \mathit{valuations}(\mathcal{S}'), R \rangle$ such that $\models^{\mathscr{M}'} \mathcal{D}^-_\Phi$. Clearly, we have $\not\models^{\mathscr{M}'} \Psi$. ■

We also give a sufficient condition for the success of a contraction.

Theorem 9.3
Let Φ *be an effect or an executability law such that* $\mathcal{S} \not\models_{\mathsf{PDL}} \Phi$. *If* $\mathcal{D}$ *is modular, then* $\mathcal{D}^-_\Phi \not\models \Phi$.

Proof:
Suppose $\mathcal{D}^-_\Phi \models \Phi$. From the fact that $\mathcal{D}$ is modular, Corollary 9.1 gives us that $\models^{\mathscr{M}'} \Phi$ for all $\mathscr{M}' \in \mathscr{M}^-_\Phi$, for some $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{D}$.

Let Φ be of the form $\varphi \rightarrow [a]\psi$, for $\varphi, \psi \in \mathfrak{Fml}$. If $\models^{\mathscr{M}'} \varphi \rightarrow [a]\psi$ for every $\mathscr{M}' \in \mathscr{M}^-_{\varphi \rightarrow [a]\psi}$, then even for $\mathscr{M}'' = \langle W, R \cup R'_a \rangle$ such that $R'_a = \{(w, w') \ : \models^{\mathscr{M}}_{w} \varphi\}$, we have $\models^{\mathscr{M}''} \varphi \rightarrow [a]\psi$. By our semantics, this is the case only if $W = \mathit{valuations}(\psi)$, in which case $\mathcal{S} \models_{\mathsf{PDL}} \varphi \rightarrow [a]\psi$.

Let now Φ have the form $\varphi \to \langle a \rangle \top$, for some $\varphi \in \mathfrak{Fml}$. If $\models^{\mathscr{M}'} \varphi \to \langle a \rangle \top$ for every $\mathscr{M}' \in \mathscr{M}^{-}_{\varphi \to \langle a \rangle \top}$, then even for $\mathscr{M}'' = \langle W, \emptyset \rangle \in \mathscr{M}^{-}_{\varphi \to \langle a \rangle \top}$, we have $\models^{\mathscr{M}''} \varphi \to \langle a \rangle \top$. But this is true only if $W = \mathit{valuations}(\neg\varphi)$, in which case $\mathcal{S} \models_{\mathsf{PDL}} \varphi \to \langle a \rangle \top$. ■

What is the status of the AGM-postulates for contraction in our framework? First, contraction of static laws satisfies all the postulates, as soon as the underlying classical contraction operator $\ominus$ satisfies all of them.

In the general case, however, our constructions do not satisfy the central postulate of preservation $\mathcal{D}^{-}_{\Phi} = \mathcal{D}$ if $\mathcal{D} \not\models \Phi$. Indeed, suppose we have a language with only one atom p, and a model $\mathscr{M}$ with two worlds $w = \{p\}$ and $w' = \{\neg p\}$ such that $wR_a w'$, $w'R_a w$, and $w'R_a w'$ (Figure 9.6). Then $\models^{\mathscr{M}} p \to [a]\neg p$ and $\not\models^{\mathscr{M}} [a]\neg p$, i.e., $\mathscr{M}$ is a model of the effect law $p \to [a]\neg p$, but not of $[a]\neg p$. Now the contraction $\mathscr{M}^{-}_{[a]\neg p}$ yields the model $\mathscr{M}'$ such that $R_a = W \times W$. Then $\not\models^{\mathscr{M}'} p \to [a]\neg p$, i.e., the effect law $p \to [a]\neg p$ is not preserved. Our contraction operation thus behaves rather like an update operation.

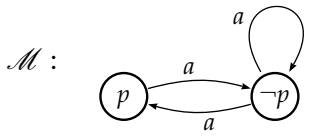

Figure 9.6: Counter-example to preservation.

Now let us focus on the other postulates. Since our operator has a behavior which is close to the update postulate, we focus on the following basic erasure postulates introduced in [66].

KM1 $Cn(\mathcal{D}^{-}_{\Phi}) \subseteq Cn(\mathcal{D})$

Postulate **KM1** does not always hold because it is possible to make the formula $\varphi \to [a]\bot$ valid in the resulting theory by removing elements of R_a (cf. Definition 9.3).

KM2 $\Phi \notin Cn(\mathcal{D}^{-}_{\Phi})$

Under the condition that $\mathcal{D}$ is modular, Postulate **KM2** is satisfied (cf. Theorem 9.3).

KM3 If $Cn(\mathcal{D}_1) = Cn(\mathcal{D}_2)$ and $\models_{\mathsf{PDL}} \Phi_1 \leftrightarrow \Phi_2$, then $Cn(\mathcal{D}_{1\Phi_2}^{-}) = Cn(\mathcal{D}_{2\Phi_1}^{-})$.

Theorem 9.4
If $\mathcal{D}_1$ and $\mathcal{D}_2$ are modular and the propositional contraction operator $\ominus$ satisfies Postulate **KM3***, then Postulate* **KM3** *is satisfied for every* PDL*-formulas Φ_1, Φ_2.*

Proof:
The proof follows straightforwardly from our results: since $Cn(\mathcal{D}_1) = Cn(\mathcal{D}_2)$ and $\models_{\overline{\mathsf{PDL}}} \Phi_1 \leftrightarrow \Phi_2$, they have, pairwise, the same models. Hence, given $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{D}_1$ and $\models^{\mathscr{M}} \mathcal{D}_2$, $\mathscr{M}_{\overline{\Phi_1}}$ and $\mathscr{M}_{\overline{\Phi_2}}$ have the same semantical operations. Because $\mathcal{D}_1$ and $\mathcal{D}_2$ are modular, Corollary 9.1 guarantees we get the same syntactical results. Moreover, as the classical contraction operation $\ominus$ satisfies Postulate **KM3**, it follows that $Cn(\mathcal{D}_{1\overline{\Phi_2}}) = Cn(\mathcal{D}_{2\overline{\Phi_1}})$. ■

9.4 Contracting Implicit Static Laws

There can be many reasons why a theory should be changed. Following the discussion in Chapter 7, here we focus on the case where it has some classical consequence φ the designer is not aware of.

If φ is taken as intuitive, then, normally, no change has to be done at all, unless we want to keep abide on the modularity principle and thus make φ explicit by adding it to $\mathcal{S}$. In the scenario example of Section 9.3, if the knowledge engineer's universe has immortal turkeys, then she would add the static law *alive* to $\mathcal{S}$.

The other way round, if φ is not intuitive, as long as φ is entailed by $\mathcal{D}$, the goal is to avoid such an entailment, i.e., what we want is $\mathcal{D}^-_\varphi \not\models \varphi$. In the mentioned scenario, the knowledge engineer considers that having immortal turkeys is not reasonable and thus decides to change the domain description.

This means that action theories that are not modular need to be changed, too. Such a changing process is driven by the problematic part of the theory detected by Algorithm 7.1.

It seems that in general implicit static laws are not intuitive. Therefore their contraction is more likely to happen than their addition.[2] In the example above, the action theory has to be contracted by *alive*.[3] In order to contract the action theory, the designer has several choices:

- Contract the set $\mathcal{S}$. (In this case, such an operation is not enough, since *alive* is a consequence of the rest of the theory.)

[2] In all the examples in which we have found implicit static laws that are intuitive they are so evident that the only explanation for not having them explicitly stated is that they have been forgotten by the theory's designer (cf. Section 7.3).

[3] Here the change operation is a revision-based operation rather than an update-based operation since we mainly "fix" the theory.

- Weaken the effect law $[tease]walking$ to $alive \rightarrow [tease]walking$, since the original effect law is too strong. This means that in a first stage the designer has to contract the theory and in a second one expand the effect laws with the weaker law. The designer will usually choose this option if she focuses on the effect preconditions of actions.

- Weaken the executability law $\langle tease\rangle\top$ by rephrasing it as $alive \rightarrow \langle tease\rangle\top$: first the executability is contracted and then the weaker one is added to the resulting set of executability laws. The designer will choose this option if she focuses on preconditions for action execution.

The analysis of this example shows that the choice of what change has to be carried out is up to the knowledge engineer. Such a task can get more complicated when ramifications are involved. To witness, suppose our scenario has been formalized as follows: $\mathcal{S} = \{walking \rightarrow alive\}$, $\mathcal{E} = \{[shoot]\neg alive\}$, $\mathcal{X} = \{\langle shoot\rangle\top\}$, and $\rightsquigarrow = \{\langle shoot, \neg alive\rangle\}$. From the corresponding action theory, we can derive the inexecutability $walking \rightarrow [shoot]\bot$ and thus the implicit static law $\neg walking$. In this case, we have to change the theory by contracting the frame axiom $walking \rightarrow [shoot]walking$ (which amounts to adding the missing indirect dependence $shoot \rightsquigarrow \neg walking$).

For an account of how elaboration tolerant our theory change method is, we refer the reader to the next chapter, where we also discuss about related work on modularity and update of domain descriptions.

Chapter 10

where we have a small talk with our neighbors

Discussion and Related Work

I am a part of all that I have seen.

— Alfred Lord Tennyson

In this chapter, we analyze whether our modularity paradigm is in line with the requirements that logical modules are expected to satisfy, and also address existing work in the literature about the meta-theory of actions. We then investigate the principle of elaboration tolerance in our theory change framework and discuss about other techniques for changing a domain description.

10.1 How Modular our Modules Are

Here we comment on the properties logical modules should have by assessing how our notion of modularity behaves with respect to them. The following criteria were compiled by Fodor [34] and Garson [38]. They also correspond to most of the design principles commonly found in software engineering.

Domain specificity: A module is domain specific if it is designed to draw conclusions over a limited domain of expertise [38]. Since with our modules we can reason in a set of laws concerning only the actions and fluents describing a sub-domain, regardless of the rest of the description, we can say that our modules are domain specific.

Accuracy: A module is accurate if it proves all sentences in its domain of application. Our modules are accurate for by satisfying the principle of modularity they can prove any formula in their respective domain that also follows from the whole theory.

Auto-sufficiency: A module should contain all the data it needs to solve problems in its domain, so that the only input it needs is the question to be answered [38]. This

also relates to accuracy and modularity. Clearly, by guaranteeing that there are no implicit laws, our modules possess all the data they need for answering a query.

Performance: Inferences in the module should be faster than in the whole description. First, because the module may use an inference relation less complex than the global one (e.g., our module of static laws uses the classical consequence relation $\models_{\overline{\mathsf{CPL}}}$). Second, the number of formulas of a module is supposed to be significantly smaller than that of the whole description, then even if algorithms with exponential complexity are used, the size of the problem is small enough to ensure practical response times [38]. With the results of Section 8.3, we achieve such an improvement in performance.

Encapsulation: We achieve encapsulation if modules do not need to access global information concerning the problem to be solved. As we have seen along this work, static laws are (by definition) laws of the world and (by definition) they must be accessed by all modules. Here we got rid of this by putting them inside each module. The price to pay is the replication of the same set of static laws in all modules.

Independence: Modules should be independent in the sense that further modifications (elaborations) of the description are carried out with as little disruption as possible. This means that additions or removals of modules should not affect the behavior of the rest of the system. This relates to the principle of elaboration tolerance [88, 89] in reasoning about actions. Despite some attempts of quantifying such an independence [3], that remains an open issue of research. Regarding our modules, we can easily see that they depend one upon the others: first, as already expected, because of the static laws. Second, because changing laws of one type is very likely to affect laws of other types, too (e.g. if we replace $\mathit{hasGun} \rightarrow \langle \mathit{shoot} \rangle \top$ in our example by just $\langle \mathit{shoot} \rangle \top$, there would be a new static law, viz. hasGun). Third, since changing a module may add implicit laws into the theory, modularity may have to be checked again. Fortunately, with the results of Section 8.3, we can guarantee independence of action laws for actions $a_1, \ldots, a_n$ from action laws that mention actions other than $a_1, \ldots, a_n$, under the condition that the added module is also itself modular [54].

10.2 Other Modularity and Consistency Notions

A Meta-theory of the Situation Calculus

Pirri and Reiter have investigated the meta-theory of the Situation Calculus [96]. In a spirit similar to ours, they use executability laws and effect laws. Contrarily to us,

their executability laws are equivalences and are thus at the same time inexecutability laws. As they restrict themselves to domains without ramifications, there are no static laws, i.e., $\mathcal{S} = \emptyset$. For this setting, they give a syntactical condition on effect laws guaranteeing that they do not interact with the executability laws in the sense that they do not entail implicit static laws. Basically, the condition says that when there are effect laws $\varphi_1 \rightarrow [a]\psi$ and $\varphi_2 \rightarrow [a]\neg\psi$, then φ_1 and φ_2 are inconsistent (which essentially amounts to having in their theories a kind of "implicit static law schema" of the form $\neg(\varphi_1 \wedge \varphi_2)$).

This then allows them to show that such theories are always consistent. Moreover, they thus simplify the entailment problem for this calculus, and show for several problems such as consistency or regression that only some of the modules of an action theory are necessary.

In the object-oriented Situation Calculus [2, 4], executabilities are as in [96] and the same condition on effect laws is assumed, which syntactically precludes the existence of implicit static laws. The frame problem is solved using Reiter's solution [100] and then is also restricted to domains without static laws. Ramifications are dealt with by compiling them away *à la* Reiter and Lin [80] based on the method given in [91], which takes into account only some restricted state constraints.

In spite of using many of the object-oriented paradigm tools and techniques, no mention is made to the concepts of cohesion and coupling [98], which are closely related to modularity [57]. In the approach presented in [2], even if modules are individually highly cohesive, they are not necessarily lowly coupled, due to the dependence between objects in the reasoning phase. We do not investigate this further here, but conjecture that this could be done there by, during the reasoning process defined for that approach, avoiding passing to a module a formula of a type different from those it contains (cf. Chapter 3).

The present work generalizes and extends Pirri and Reiter's result to the case where $\mathcal{S} \neq \emptyset$ and both these works where the syntactical restriction on effect laws is not made. It also constitutes a better approach for domains with ramifications as we do not impose any restriction on the domain constraints we can deal with.

Moreover, by guaranteeing satisfaction of modularity, our domain descriptions can be decomposed according to the ideas in [2]. We illustrate this with the example from Section 3.2:

$$\mathcal{D}_1 = \langle \mathcal{L}_1, \models_{\overline{\mathsf{CPL}}}, \langle \{walking_1 \rightarrow alive_1\}, \{walking_1, alive_1\}\rangle\rangle$$

$$\mathcal{D}_2 = \langle \mathcal{L}_2, \models_{\mathsf{PDL}}, \langle \left\{ \begin{array}{c} alive_2 \rightarrow \langle tease_2 \rangle \top, \\ hasGun_2 \rightarrow \langle shoot_2 \rangle \top \end{array} \right\}, \{alive_2\} \rangle \rangle$$

$$\mathcal{D}_3 = \langle \mathcal{L}_3, \models_{\rightsquigarrow}, \langle \left\{ \begin{array}{c} \neg loaded_3 \rightarrow [load_3] loaded_3, \\ loaded_3 \rightarrow [shoot_3] \neg alive_3, \\ [tease_3] walking_3 \end{array} \right\}, \{walking_3, alive_3\} \rangle \rangle$$

$$\mathcal{D}_4 = \langle \mathcal{L}_4, \models_{\mathsf{PDL}}, \langle \left\{ \begin{array}{c} \neg hasGun_4 \rightarrow [shoot_4] \bot, \\ \neg alive_4 \rightarrow [tease_4] \bot \end{array} \right\}, \{alive_4\} \rangle \rangle$$

(For this example, we assume we have detected all implicit laws of the description in Section 3.2 with Algorithm 7.1 and then contracted the theory by the unintuitive static law *alive*.)

Hence with our approach we have the advantage of a more expressive power, as we can reason about inexecutabilities, and a better modularity in the sense that we do not combine formulas that are conceptually different (viz. executabilities and inexecutabilities). Moreover, by guaranteeing nonexistence of implicit laws, many of the results presented in the referred work, e.g., conditional independence (cf. Section 3.2), transfer to ours.

Consistency in the Presence of Ramifications

Zhang *et al.* [118] have also proposed an assessment of what a good action theory should look like. They develop the ideas in the framework of EPDL [119], an extended version of PDL which allows for propositions as modalities to represent a causal connection between literals (cf. Section 6.5). We do not present the details of that, but concentrate on the main meta-theoretical results.

Zhang *et al.* propose a normal form for describing action theories,[1] and investigate three levels of consistency. Roughly speaking, a set of laws $\mathcal{T}$ is *uniformly consistent* if it is globally consistent (i.e., $\mathcal{T} \not\models_{\mathsf{EPDL}} \bot$); a formula Φ is $\mathcal{T}$*-consistent* if $\mathcal{T} \not\models_{\mathsf{EPDL}} \neg\Phi$, for $\mathcal{T}$ a uniformly consistent theory; $\mathcal{T}$ is *universally consistent* if (in our terms) every logically possible world is accessible.

Furthermore, two assumptions are made to preclude the existence of implicit qualifications. Satisfaction of such assumptions means the theory under consideration is

[1] But not as expressive as one might think: For instance, in modeling the nondeterministic action of dropping a coin on a chessboard, we are not able to state $[drop](black \vee white)$. Instead, we should write something like $[drop_{black}]black$, $[drop_{white}]white$, $[drop_{black,white}]black$ and $[drop_{black,white}]white$, where $drop_{black}$ is the action of dropping the coin on a black square (analogously for the others) and $drop = drop_{black} \cup drop_{white} \cup drop_{black,white}$, with "$\cup$" the nondeterministic composition of actions.

safe, i.e., it is uniformly consistent. Such a normal form justifies the two assumptions made and on which their notion of good theories relies.

Given this, they propose algorithms to test the different versions of consistency for a theory $\mathcal{T}$ that is in normal form. This test essentially amounts to checking whether $\mathcal{T}$ is *safe*, i.e., whether $\mathcal{T} \models_{\overline{\mathsf{EPDL}}} \langle a \rangle \top$, for every action a. Success of this check should mean that the theory under analysis satisfies the consistency requirements.

Although they are concerned with the same kind of problems that have been discussed in this work, they take an overall view of the subject, in the sense that all problems are dealt with together. This means that in their approach no special attention (in our sense) is given to the different components of the theory, and then every time something is wrong with it this is taken as a global problem inherent to the theory as a whole. Whereas such a "systemic" view of action theories is not necessarily a drawback (we have just seen the strong interaction that exists between the different sets of laws composing an action theory), being modular in our sense allows us to better identify the "problematic" laws and take care of them. Moreover, the advantage of allowing to find the set of laws which must be modified in order to achieve the desired consistency is made evident by the algorithms we have proposed (while their results only allow to decide whether a given theory satisfies some consistency requirement).

Consistency and Executability

Lang *et al.* [72] address consistency of action theories in a version of the causal laws approach [83], focusing on the computational aspects.

To solve the frame problem, they suppose an abstract notion of completion. Given a theory $\mathcal{T}^a$ containing logical information about a's direct effects as well as the indirect effects that may follow (expressed in the form of causal laws), the completion of $\mathcal{T}^a$, roughly speaking, is the original theory $\mathcal{T}^a$ amended of some axioms stating the persistence of all non-affected (directly nor indirectly) literals. (Note that such a notion of completion is close to the underlying semantics of the dependence relation used throughout the present work, which essentially amounts to the explanation closure assumption [102].)

Their EXECUTABILITY problem is to check whether action a is executable in all possible initial states (Zhang *et al.*'s safety property). This amounts to testing whether every possible state w has a successor w' reachable by a such that w and w' both satisfy the completion of $\mathcal{T}^a$. For the Walking Turkey Scenario, the formalization of action *tease* with causal laws is given by:

$$\mathcal{T}^{tease} = \left\{ \begin{array}{c} \top \overset{tease}{\Rightarrow} walking, \\ \neg alive \Rightarrow \neg walking \end{array} \right\}$$

where the first formula is a conditional effect law for *tease*, and the latter a causal law in McCain and Turner's sense (cf. Section 6.3). We will not dive in the technical details, and just note that the executability check will return "no" for this example as *tease* cannot be executed in a state satisfying $\neg alive$.

In the mentioned work, the authors are more concerned with the complexity analysis of the problem of doing such a consistency test and no algorithm for performing it is given, however. In spite of the fact their motivation is the same as ours, again what is presented is a kind of "yes-no tool" which can help in doing a meta-theoretical analysis of a given action theory, and many of the comments concerning Zhang *et al.*'s approach could be repeated here.

Another criticism that could be made about both these approaches concerns the assumption of full executability they rely on. We find it too strong to require all actions to be always executable (cf. Section 8.2), and to reject as bad an action theory admitting situations where some action cannot be executed at all. As an example, consider a very simple action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$, where $\mathcal{S} = \{walking \rightarrow alive\}$, $\mathcal{E} = \{[tease]walking\}$, $\mathcal{X} = \{\langle tease \rangle \top\}$, $\mathcal{I} = \emptyset$, and $\rightsquigarrow = \{\langle tease, walking \rangle\}$. Observe that, with our approach, it suffices to derive the implicit inexecutability law $\neg alive \rightarrow [tease]\bot$, change $\mathcal{I}$, and the system will properly run in situations where $\neg alive$ is the case.

On the other hand, if we consider the equivalent representation of such an action theory in the approach of Lang *et al.*, after computing the completion of $\mathcal{T}^{tease}$, if we test its executability, we will get the answer "no", the reason being that *tease* is not executable in the possible state where $\neg alive$ holds. Such an answer is correct, but note that with only this as guideline we have no idea about where a possible modification in the action theory should be carried out in order to achieve full executability for *tease*. The same observation holds for Zhang *et al.*'s proposal.

Just to see how things can be even worse, let $\mathcal{D}'$ be the same action theory as above, but with $\mathcal{X} = \{alive \rightarrow \langle tease \rangle \top\}$, obtained by the correction of $\mathcal{D}$ above with the algorithms we proposed. Observe that $\mathcal{D}'$ satisfies all our postulates. It is not hard to see, however, that the representation of such an action theory in the above frameworks, when checked by their respective consistency tests, is still considered to have a problem.

This problem arises because Lang *et al.*'s proposal do not allow for executability

laws, thus one cannot make the distinction between $\mathcal{X} = \{\langle tease \rangle \top\}$, $\mathcal{X} = \{alive \rightarrow \langle tease \rangle \top\}$ and $\mathcal{X} = \emptyset$. By their turn, Zhang *et al.*'s allows for specifying executabilities, however their consistency definitions do not distinguish the cases $alive \rightarrow \langle tease \rangle \top$ and $\langle tease \rangle \top$.

Modular Action Languages

Lifschitz and Ren [77] propose an action description language derived from $\mathcal{C}+$ [45] in which action theories can also be decomposed in modules. Contrarily to our setting, in theirs a module is not a set of formulas for given action a, but rather a description of a subsystem of the theory, i.e., each module describes a set of interrelated fluents and actions (cf. Section 3.2). As an example, a module describing Lin's suitcase scenario [78] should contain all causal laws in the sense of $\mathcal{C}+$ that are relevant to the scenario. Actions or fluents having nothing to do, neither directly nor indirectly, with the suitcase should be described in different modules. This feature makes such a decomposition somewhat domain-dependent, while here we have proposed a type-oriented modularization of the formulas, which does not depend on the domain.

In the referred work, modules can be defined in order to specialize other modules. This is done by making the new module to inherit and then specialize other modules' components. This is an important feature when elaborations are involved. In the suitcase example, adding a new action relevant to the suitcase description can be achieved by defining a new module inheriting all properties of the old one and containing the causal laws needed for the new action. Such ideas are interesting from the standpoint of software and knowledge engineering: reusability is an intrinsic property of the framework, and easy scalability promotes elaboration tolerance.

Consistency of a given theory and how to prevent (independent or inherited) conflicts between modules however is not addressed.

Other Logics

A concept similar to that of implicit static laws was firstly addressed, as far as we are concerned, in the realm of regulation consistency with deontic logic [17]. Indeed, the notions of regulation consistency given in the mentioned work and that of modularity presented in [58] and used here can be proved to be equivalent. The main difference between the mentioned work and the approach in [58] relies on the fact that in [17] some syntactical restrictions on the formulas have to be made in order to make the algorithm that is proposed to work.

In [22] an algorithm is proposed to, given a monolithic description of a web ontology in description logic [7], find a good modularization according to criteria similar to Garson's. As we have seen, those are stronger than our notion of modularity in the sense that a given formula should be derivable only from a single module (cf. Section 3.3). This means that applying the method in [22] to reasoning about actions would preclude the natural overlapping between modules, that is inherent in reasoning about actions theories. Moreover, modules defined in that way are sub-domain oriented and may have any type of formula, what can still difficult the module's maintainability.

A different approach of the work we presented here can be found in [57], where modularity of action theories is assessed from a software engineering perspective in the Situation Calculus.

Based on the results we have seen in Chapter 4, in [59] we have defined a modularity approach for description logic [7]. Such a notion of modularity we present there is related to uniform interpolation for TBoxes [41]. Let $\mathit{concepts}(\mathcal{T})$ denote the concept names and $\mathit{roles}(\mathcal{T})$ the role names occurring in a TBox $\mathcal{T}$. Given $\mathcal{T}$ and a signature $S \subseteq \mathit{concepts}(\mathcal{T}) \cup \mathit{roles}(\mathcal{T})$, a TBox $\mathcal{T}^S$ over $(\mathit{concepts}(\mathcal{T}) \cup \mathit{roles}(\mathcal{T})) \setminus S$ is a *uniform interpolant* of $\mathcal{T}$ outside S if and only if:

- $\mathcal{T} \models \mathcal{T}^S$;
- $\mathcal{T}^S \models C \sqsubseteq D$ for every $C \sqsubseteq D$ that has no occurrences of symbols from S.

(Here, $\models$ denotes the entailment for description logics.) It is not difficult to see that a partition $\{\mathcal{T}^\emptyset\} \cup \{\mathcal{T}^{R_i} \ : \ R_i \in \mathit{roles}(\mathcal{T})\}$ is modular if and only if every $\mathcal{T}^{R_i}$ is a uniform interpolant of $\mathcal{T}$ outside $\mathit{roles}(\mathcal{T}) \setminus \{R_i\}$. In [111] there are complexity results for computing uniform interpolants in $\mathcal{ALC}$.

Still in the realm of description logics, in [41] a notion of conservative extension is defined that is similar to our modularity. There, $\mathcal{T}_1 \cup \mathcal{T}_2$ is a *conservative extension* of $\mathcal{T}_1$ if and only if for all concepts C, D built from $\mathit{concepts}(\mathcal{T}_1) \cup \mathit{roles}(\mathcal{T}_1)$, $\mathcal{T}_1 \cup \mathcal{T}_2 \models C \sqsubseteq D$ implies $\mathcal{T}_1 \models C \sqsubseteq D$.

Given our Theorem 4.1, we can show that checking for modularity can be reduced to checking for conservative extensions of $\mathcal{T}^\emptyset$. Indeed, supposing that the signature of $\mathcal{T}^\emptyset$ is the set of all concept names, we have that $\mathcal{T}$ is modular if and only if for every role R_i, $\mathcal{T}^{R_i} \cup \mathcal{T}^\emptyset$ is a conservative extension of $\mathcal{T}^\emptyset$.

10.3 How Elaboration Tolerant We Are

The principle of elaboration tolerance has been proposed by McCarthy [88]. Roughly, it states that the effort required to add new information to a given representation (new laws or entities) should be proportional to the complexity of the information being added, i.e., it should not require the complete reconstruction of the old theory [105].

Since then, many formalisms claim, in a more or less tacit way, to satisfy such a principle. Nevertheless, for all this time there has been a lack of good formal criteria allowing for the evaluation of theory change difficulty and, consequently, comparisons between different frameworks are carried out in a subjective way.

The proposal by Amir [3] made the first steps in formally answering what difficulty of changing a theory means by formalizing one aspect of elaboration tolerance. The basic idea is as follows: let $\mathcal{D}_0$ be the original domain description and let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two equivalent (and different) descriptions such that each one results from $\mathcal{D}_0$ by the application of some sequence of operations (additions and/or deletions of formulas). The resulting theory whose transformation from $\mathcal{D}_0$ has the shortest length (number of operations) is taken as the most elaboration tolerant.

Nevertheless, in the referred work only addition/deletion of *axioms* is considered, i.e., changes in the logical language or contraction of consequences of the theory not explicitly stated in the original set of axioms are not taken into account. This means that even the formal setting given in [3] is not enough to evaluate the difficulty of theory change in a broad sense. Hence the community still needs formal criteria that allow for the comparison between more complex changes carried out by frameworks like ours, for example.

Of course, how elaboration tolerant a given update/revision method is strongly depends on its underlying formalism for reasoning about actions, i.e., its logical background, the solution to the frame problem it implements, the hypotheses it relies on, etc. In what follows, we discuss how the dependence-based approach here used behaves when expansion is considered. Most of the comments concerning consequences of expansion can also be stated for contraction. We do that with respect to some of the qualitative criteria given in [89]. In all that follows, we suppose that the resulting theory is consistent.

Adding effect laws: In the dependence-based framework, adding the new effect law $\varphi \rightarrow [a]\psi$ to the theory demands a change in the dependence relation $\rightsquigarrow$, and hence it means changing the consequence relation in $\mathcal{D}$. In that case, the maximum number of

statements added to $\rightsquigarrow$ is $card(\{\ell : \ell \in \chi, \text{ for all } \chi \in NewCons(\psi, \mathcal{S})\})$ (dependences for all indirect effects have to be stated, too). This is due to the explanation closure nature of the reasoning behind dependence (for more details, see [14]). Because of this, according to Shanahan [105], explanation closure approaches are not elaboration tolerant when dealing with the ramification problem. In order to achieve that, the framework should have a mechanism behaving like circumscription that automatically deals with ramifications. This raises the question: "if we had an automatic (or even semi-automatic) procedure to do the job of generating the indirect dependences, could we say the framework is elaboration tolerant?". We think we can answer positively to such a question, since we can semi-automatically generate the dependence relation from a set of effect laws with the method in [13].

Adding executability laws: Such a task demands only a change in the set $\mathcal{X}$ of executabilities, possibly introducing implicit static laws as a side effect.

Adding static laws: Besides expanding the set $\mathcal{S}$, adding new (indirect) dependences may be required, changing the consequence relation component of the domain description (see above).

Adding frame axioms: If the frame axiom $\neg\ell \rightarrow [a]\neg\ell$ has to be valid in the resulting theory, expunging the dependence $a \rightsquigarrow \ell$ should do the job, which in our case means a change in the consequence relation.

Adding a new action name: Without loss of generality we can assume the action in question was already in the language. In that case, we expect just to add effect or executability laws for it. For the former, at most $card(\mathfrak{Lit})$ dependences will be added to $\rightsquigarrow$. (We point out nevertheless that the requirement made in [89] that the addition of an action irrelevant for a given plan in the old theory should not preclude it in the resulting theory is too strong. Indeed, it is not difficult to imagine a new action forcing an implicit static law from which an inexecutability for some action in the plan can be derived. The same holds for the item below.)

Adding a new fluent name: In the same way, we can suppose the fluent was already in the language. Such a task amounts thus to one or more of the above expansions. There will be at most $2 \times card(\mathfrak{Act})$ new elements added to $\rightsquigarrow$.

Because of forcing formulas to be explicitly stated in their respective modules (and thus possibly making them inferable from two or more different modules at once), intuitively modularity could be seen to diminish elaboration tolerance. For instance,

when contracting a classical formula φ from a non-modular theory, it seems reasonable to expect not to change the set of static laws $\mathcal{S}$, while the theory being modular surely forces changing such a module. However it is not difficult to conceive non-modular theories in which contraction of a formula φ may demand a change in $\mathcal{S}$ as well. To witness, let $\mathcal{S} = \{\varphi_1 \rightarrow \varphi_2\}$ in an action theory from whose dynamic part we (implicitly) infer $\neg\varphi_2$. Then, contracting $\neg\varphi_1$ keeping $\neg\varphi_2$ would necessarily ask for a change in $\mathcal{S}$. We point out nevertheless that, in both cases (modular and non-modular), the extra work in changing other modules stays in the mechanical level, i.e., in the machinery that carries out the theory modification, and does not augment in a significant way the amount of work the knowledge engineer is expected to do.

10.4 Other Update Methods

Following [73, 75], Eiter *et al.* [31] have investigated update of action domain descriptions. They define a version of action theory update in an action language and give complexity results showing how hard such a task can be.

Update of action descriptions in their sense is always relative to some conditions (interpreted as knowledge possibly obtained from earlier observations and that should be kept). This characterizes a constraint-based update. In the example they give, change must be carried out preserving the assumption that pushing the button of the remote control is always executable. Actually, the method is more subtle, as new effect laws are added constrained by the *addition* of viz. an executability law for the new action under concern. In the example, the constraint (executability of push) was not in the original action description and must figure in the updated theory.

They describe domains of actions in a fragment of the action language $\mathcal{C}$ [40]. However they do not specify which fragment, so it is not clear whether the claimed advantages $\mathcal{C}$ has over $\mathcal{A}$ really transfer to their framework. At one hand, their approach deals with indirect effects, but they do not talk about updating a theory by a law with a nondeterministic action.

Eiter *et al.* consider a theory $\mathcal{T}$ as comprising two main components: $\mathcal{T}_u$, the part of the theory that must remain unchanged, and $\mathcal{T}_m$, the part concerning the statements that are allowed to change. The crucial information to the associated solution to the frame problem is always in $\mathcal{T}_u$.

Given a theory $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_m$, $((\mathcal{T}_u \cup \mathcal{T}_m), \mathcal{T}', C)$ is the problem of updating $\mathcal{T}$ by $\mathcal{T}' \subseteq \mathcal{S} \cup \mathcal{E}$ warranting the result satisfies all constraints in $C \subseteq \mathcal{S} \cup \mathcal{X}$.

Even though they do not explicitly state postulates for their kind of theory update, they establish conditions for the update operator to be successful. Basically, they claim for consistency of the resulting theory; maintenance of the new knowledge and the invariable part of the description; satisfaction of the constraints in C; and also minimal change.

In some examples that they develop, the illustrated "partial solution" does not satisfy C due to the existence of implicit laws (cf. their Example 1, where there is an implicit inexecutability law). To achieve a solution, while keeping C, some other laws must be dropped (in the example, the agent gives up a static law).[2]

Just to see the link between update by subsumed laws and addition of implicit static laws, we note that their Proposition 1 is the same as our Corollary 7.1: every implicit static law in our sense is trivially a subsumed law in Eiter *et al.*'s sense.

With their method, we can also contract by a static and an effect law. Contraction of executabilities are not explicitly addressed, and weakening (replacing a law by a weaker one) is left as future work.

[2]This does not mean however that the updated theory will necessarily contain no implicit law.

Chapter 11

where we finish to start

Conclusion

Not every end is a goal. The end of a melody is not its goal; however, if the melody has not reached its end, it would also not have reached its goal. A parable.

— Nietzsche

Our contribution is twofold: general, as we presented postulates that apply to all reasoning about actions formalisms; and specific, as we proposed algorithms for a dependence-based solution to the frame problem.

We have identified and made a critique of the main approaches of logical modularity for domain descriptions, pointing out their characteristics and showing why they do not completely assess modularity in the sense descriptions in reasoning about actions need.

We have argued that modularity as commonly used in programming or defined in works on formal logic are not appropriate in reasoning about actions. In the first case because of expressivity restrictions. In the second because modularity of logical theories are usually too strong and shows to be of no much aid if the theory is a description of a scenario in reasoning about actions

We have analyzed the principle of modularity for logics in general defined by Garson. Such a notion of modularity as defined in [38] and adopted in [22] can be reduced to the concepts of *cohesion* and *coupling* [108, 98] in software engineering. In [57] we have seen the difficulty of requiring a domain description in reasoning about actions to satisfy these two principles.

The main motivation in the original work by Garson is the intractability of consistency check in classical first-order logic. That is the reason he moves to relevant logic

in order to get rid of the principle of explosion and hence get a formal substratum in which descriptions fit better with his notion of local completeness.

The principle of explosion is not a reason on its own to abandon classical logic. We agree with Cuenca Grau and colleagues [22] when they say that we can turn our attention to consistent theories and give an account of modularity even in the presence of the principle of explosion. Since our aim is to point out whether a theory is good or not, if it is inconsistent, then it simply cannot be good. Moreover, we focus on how to refine modularity, not on how to force modularity to hold for inconsistent theories. We have shown that, despite the principle of explosion, and with some amendments, we can have a good account of modularity for theories in reasoning about actions.

We have defined here our concept of modularity of an action theory and pointed out some of the problems that arise if it is not satisfied. In particular we have argued that the non-dynamic part of action theories could influence but should not be influenced by the dynamic one.[1]

We have put forward some postulates, and in particular tried to demonstrate that when there are implicit static and inexecutability laws then one has slipped up in designing the action theory in question. As shown, a possible solution comes into its own with Algorithms 7.1 and 7.2, which can give us some guidelines in correcting an action theory if needed. By means of examples, we have seen that there are several alternatives of correction, and choosing the right module to be modified as well as providing the intuitive information that must be supplied is a task that is up to the knowledge engineer.

Given the difficulty of exhaustively enumerating all the preconditions under which a given action is executable (and also those under which such an action cannot be executed), it is reasonable to expect that there is always going to be some executability precondition φ_1 and some inexecutability precondition φ_2 that together lead to a contradiction, forcing, thus, an implicit static law $\neg(\varphi_1 \wedge \varphi_2)$. This is the reason we propose to state some information about both executabilities and inexecutabilities, and then run the algorithms in order to improve the description.

It could be argued that unintuitive consequences in action theories are mainly due to badly written axioms and not to the lack of modularity. True enough, but what we have presented here is the case that making a domain description modular gives us

[1] It might be objected that it is only by doing experiments that one learns the static laws that govern the universe. But note that this involves *learning*, whereas here – as always done in the reasoning about actions field – the static laws are known once forever, and do not evolve.

a tool to detect at least some of such problems and correct it. (But note that we do not claim to correct badly written axioms automatically and once for all). Besides this, having separate entities in the ontology and controlling their interaction help us to localize where the problems are, which can be crucial for real world applications.

In this work we have illustrated by some examples what we can do in order to make a theory intuitive. This involves theory modification. We have presented a general method for changing a domain description given a formula we want to contract.

We have defined a semantics for theory contraction and also presented its syntactical counterpart through contraction operators. Soundness and completeness of such operators with respect to the semantics have been established (Corollary 9.1).

We have also shown that modularity is a sufficient condition for contraction to be successful (Theorem 9.3). This gives further evidence that the notion of modularity is fruitful.

Modularity is not necessarily a property of the underlying logical formalism. It is rather a property of descriptions written in such a formalism. The choice of which logical background to use in formalizing a domain may more or less ease the satisfaction of modularity.

In this work we used a weak version of PDL, but our notions and results can be applied to other frameworks as well. It is worth noting however that for first-order based frameworks the consistency checks of Algorithms 7.1 and 7.2 are undecidable. We can get rid of this by assuming that $\mathcal{D}$ is finite and that there is no function symbol in the language. In this way, the result of $NewCons(.)$ is finite and the algorithms terminate.

The dependence-based framework we have used here is a simple yet powerful account to the frame and ramification problems, within which Reiter's regression technique can be applied [26]. We have shown that regression does not necessarily build on Successor State Axioms as in Reiter's original theory, which involves quantification. Moreover, the dependence-based framework has the advantage of having a decision procedure in terms of tableau systems [12, 11] (while the Situation Calculus contains second-order axioms and is a priori not even semi-decidable).

We have also presented an example of a scenario having actions with both indeterminate and indirect effects, which leads to counterintuitive results when formalized in fluent-indexed approaches. The analysis we have carried out supports the thesis that causality should be action-indexed.

The problem with such a causal notion is that one must in some way relate actions and their indirect effects. Nevertheless, the present work is a step toward a solution to the problem of indirect dependences: indeed, if the indirect dependence $\mathit{shoot} \rightsquigarrow \neg \mathit{walking}$ is not in $\rightsquigarrow$, then after running Algorithm 7.2 we get an implicit inexecutability $(\mathit{loaded} \wedge \mathit{walking}) \rightarrow [\mathit{shoot}]\bot$, i.e., *shoot* cannot be executed if $\mathit{loaded} \wedge \mathit{walking}$ holds. Such an unintuitive inexecutability is not in $\mathcal{I}$ and thus indicates the missing indirect dependence. The general case is nevertheless more complex, and it seems that such indirect dependences cannot be computed automatically in the case of indeterminate effects.

A topic for further investigations could be considering the notion of *coherence* defined in [71] as a guideline for "repairing" a given theory. Roughly, given an action theory $\mathcal{D}$ and an *unintuitive* implicit static law φ, the formulas in $\mathcal{T}$ that are most likely to be revised are exactly those whose utility, in Kwok *et al.*'s sense, for deriving φ are the highest.

Our postulates do not take into account causality statements linking propositions such as in [78, 83], nor the qualification problem. This could be a topic for further investigation.

Bibliography

[1] E. Amir. Object-oriented first-order logic. *Electronic Transactions on Artificial Intelligence*, 3:63–84, 1999.

[2] E. Amir. (De)composition of situation calculus theories. In *Proc. 17th Natl. Conf. on Artificial Intelligence (AAAI'2000)*, pages 456–463, Austin, 2000. AAAI Press/MIT Press.

[3] E. Amir. Toward a formalization of elaboration tolerance: Adding and deleting axioms. In M.-A. Williams and H. Rott, editors, *Frontiers of Belief Revision*. Kluwer, 2000.

[4] E. Amir. Projection in decomposed situation calculus. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proc. 8th Intl. Conf. on Knowledge Representation and Reasoning (KR'2002)*, pages 315–326, Toulouse, 2002. Morgan Kaufmann Publishers.

[5] E. Amir and S. McIlraith. Partition-based logical reasoning. In Cohn et al. [20], pages 389–400.

[6] E. Amir and S. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artificial Intelligence*, 162(1–2):49–88, 2005.

[7] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *Description Logic Handbook*. Cambridge University Press, 2003.

[8] F. Baader and W. Nutt. Basic description logics. In Baader et al. [7], chapter 2, pages 47–100.

[9] A.B. Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49(1–3):5–23, 1991.

[10] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001.

[11] M. Castilho. *Modèles logiques pour le raisonnement sur les actions*. PhD thesis, Université Paul Sabatier, Toulouse, 1998.

[12] M. Castilho, L. Fariñas del Cerro, O. Gasquet, and A. Herzig. Modal tableaux with propagation rules and structural rules. *Fundamenta Informaticae*, 32(3–4):281–297, 1997.

[13] M. Castilho, O. Gasquet, and A. Herzig. A dependence-based framework for actions with indeterminate and indirect effects. Technical Report RT–98-04–R, Institut de recherche en informatique de Toulouse (IRIT), Université Paul Sabatier, February 1998. `http://www.irit.fr/LILaC/`.

[14] M. Castilho, O. Gasquet, and A. Herzig. Formalizing action and change in modal logic I: the frame problem. *J. of Logic and Computation*, 9(5):701–735, 1999.

[15] M. Castilho, A. Herzig, and I. Varzinczak. It depends on the context! a decidable logic of actions and plans based on a ternary dependence relation. In S. Benferhat and E. Giunchiglia, editors, *Workshop on Nonmonotonic Reasoning (NMR'02)*, pages 343–348, Toulouse, 2002.

[16] B. Chellas. *Modal logic: An introduction*. Cambridge University Press, 1980.

[17] L. Cholvy. Checking regulation consistency by using SOL-resolution. In *Proc. 7th Intl. Conf. on AI and Law*, pages 73–79, Oslo, 1999.

[18] K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logics and Databases*, pages 293–322. Plenum Press, New York, 1978.

[19] A. Cohn, L. Schubert, and S. Shapiro, editors. *Proc. 6th Intl. Conf. on Knowledge Representation and Reasoning (KR'98)*, Trento, 1998. Morgan Kaufmann Publishers.

[20] T. Cohn, F. Giunchiglia, and B. Selman, editors. *Proc. 7th Intl. Conf. on Knowledge Representation and Reasoning (KR'2000)*, Breckenridge, 2000. Morgan Kaufmann Publishers.

[21] W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *J. of Symbolic Logic*, 22:250–268, 1957.

[22] B. Cuenca Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Modularity and web ontologies. In Doherty et al. [29], pages 198–208.

[23] G. De Giacomo and M. Lenzerini. PDL-based framework for reasoning about actions. In M. Gori and G. Soda, editors, *Proc. 4th Congresss of the Italian Association for Artificial Intelligence (IA*AI'95)*, number 992 in LNAI, pages 103–114. Springer-Verlag, 1995.

[24] R. Demolombe. Belief change: from situation calculus to modal logic. In G. Brewka and P. Peppas, editors, *Workshop on Nonmonotonic Reasoning, Action and Change*, 2003.

[25] R. Demolombe, A. Herzig, and I. Varzinczak. Regression in modal logic. In *Methods for Modalities 2003 (M4M'03)*, Nancy, 2003.

[26] R. Demolombe, A. Herzig, and I. Varzinczak. Regression in modal logic. *J. of Applied Non-Classical Logics (JANCL)*, 13(2):165–185, 2003.

[27] P. Doherty, W. Łukaszewicz, and E. Madalinska-Bugaj. The PMA and relativizing change for action update. In Cohn et al. [19], pages 258–269.

[28] P. Doherty, W. Łukaszewicz, and A. Szałas. Explaining explanation closure. In *Proc. 9th Intl. Symposium on Methodologies for Intelligent Systems*, number 1079 in LNCS, Zakopane, Poland, 1996. Springer-Verlag.

[29] P. Doherty, J. Mylopoulos, and C. Welty, editors. *Proc. 10th Intl. Conf. on Knowledge Representation and Reasoning (KR'2006)*, Lake District, 2006. Morgan Kaufmann Publishers.

[30] J.M. Dunn. Relevance logic and entailment. In D. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, pages 117–224. D. Reidel, Dordrecht, 1986.

[31] T. Eiter, E. Erdem, M. Fink, and J. Senko. Updating action domain descriptions. In Kaelbling and Saffiotti [63], pages 418–423.

[32] J. Finger. *Exploiting constraints in design synthesis*. PhD thesis, Stanford University, Stanford, 1987.

[33] M. Fitting. *Proof methods for modal and intuitionistic logics*. D. Reidel, Dordrecht, 1983.

[34] J. Fodor. *The modularity of mind*. MIT Press, Cambridge, MA, 1983.

[35] N. Foo and D. Zhang. Dealing with the ramification problem in Extended Propositional Dynamic Logic. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 3, pages 173–191. World Scientific, 2002.

[36] K. Forbus. Introducing actions into qualitative simulation. In N. Sridharan, editor, *Proc. 11th Intl. Joint Conf. on Artificial Intelligence (IJCAI'89)*, pages 1273–1278, Detroit, 1989. Morgan Kaufmann Publishers.

[37] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, Cambridge, MA, 1988.

[38] J. Garson. Modularity and relevant logic. *Notre Dame J. of Formal Logic*, 30(2):207–223, 1989.

[39] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17(2/3&4):301–321, 1993.

[40] M. Gelfond and V. Lifschitz. Action languages. *Electronic Transactions on Artificial Intelligence*, 2(3–4):193–210, 1998.

[41] S. Ghilardi, C. Lutz, and F. Wolter. Did I damage my ontology? A case for conservative extensions in description logic. In Doherty et al. [29], pages 187–197.

[42] M. Ginsberg and D. Smith. Reasoning about actions II: The qualification problem. *Artificial Intelligence*, 35(3):311–342, 1988.

[43] L. Giordano, A. Martelli, and C. Schwind. Ramification and causality in a modal action logic. *J. of Logic and Computation*, 10(5):625–662, 2000.

[44] E. Giunchiglia, G. Kartha, and V. Lifschitz. Representing action: indeterminacy and ramifications. *Artificial Intelligence*, 95(2):409–438, 1997.

[45] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.

[46] J. Gustafsson and J. Kvarnström. Elaboration tolerance through object-orientation. *Artificial Intelligence*, 153(1–2):239–285, 2004.

[47] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logics, and the frame problem. In T. Kehler and S. Rosenschein, editors, *Proc. 5th Natl. Conf.*

on Artificial Intelligence (AAAI'86), pages 328–333, Philadelphia, 1986. Morgan Kaufmann Publishers.

[48] S. Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer Academic Publishers, 1999.

[49] D. Harel. Dynamic logic. In D. Gabbay and F. Günthner, editors, *Handbook of Philosophical Logic*, volume II, pages 497–604. D. Reidel, Dordrecht, 1984.

[50] D. Harel, J. Tiuryn, and D. Kozen. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[51] A. Herzig, L. Perrussel, and I. Varzinczak. Contracting TBoxes: the importance of being modular. Technical Report RR–2006-24-FR, Institut de recherche en informatique de Toulouse (IRIT), Université Paul Sabatier, October 2006. `http://www.irit.fr/LILaC/`.

[52] A. Herzig, L. Perrussel, and I. Varzinczak. Elaborating domain descriptions. In G. Brewka, S. Coradeschi, A. Perini, and P. Traverso, editors, *Proc. 17th Eur. Conf. on Artificial Intelligence (ECAI'06)*, pages 397–401, Riva del Garda, 2006. IOS Press.

[53] A. Herzig and O. Rifi. Propositional belief base update and minimal change. *Artificial Intelligence*, 115(1):107–138, 1999.

[54] A. Herzig and I. Varzinczak. Metatheory of actions: beyond consistency. Accepted to Artificial Intelligence Journal.

[55] A. Herzig and I. Varzinczak. An assessment of actions with indeterminate and indirect effects in some causal approaches. Technical Report 2004–08–R, Institut de recherche en informatique de Toulouse (IRIT), Université Paul Sabatier, May 2004. `http://www.irit.fr/LILaC/`.

[56] A. Herzig and I. Varzinczak. Domain descriptions should be modular. In R. López de Mántaras and L. Saitta, editors, *Proc. 16th Eur. Conf. on Artificial Intelligence (ECAI'04)*, pages 348–352, Valencia, 2004. IOS Press.

[57] A. Herzig and I. Varzinczak. Cohesion, coupling and the meta-theory of actions. In Kaelbling and Saffiotti [63], pages 442–447.

[58] A. Herzig and I. Varzinczak. On the modularity of theories. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic*, volume 5, pages 93–109. King's College Publications, 2005. Selected papers of AiML 2004 (also available at `http://www.aiml.net/volumes/volume5`).

[59] A. Herzig and I. Varzinczak. A modularity approach for a fragment of $\mathcal{ALC}$. In M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, editors, *Proc. 10th Eur. Conf. on Logics in Artificial Intelligence (JELIA'2006)*, number 4160 in LNAI, pages 216–228. Springer-Verlag, 2006.

[60] G. Hughes and M. Cresswell. *An introduction to modal logic*. Methuen & Co. Ltd, London, 1968.

[61] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence*, 56(2–3):301–353, 1992.

[62] Y. Jin and M. Thielscher. Iterated belief revision, revised. In Kaelbling and Saffiotti [63], pages 478–483.

[63] L. Kaelbling and A. Saffiotti, editors. *Proc. 19th Intl. Joint Conf. on Artificial Intelligence (IJCAI'05)*, Edinburgh, 2005. Morgan Kaufmann Publishers.

[64] A. Kakas, L. Michael, and R. Miller. $\mathcal{M}$odular-$\mathcal{E}$: an elaboration tolerant approach to the ramification and qualification problems. In C. Baral, G. Greco, N. Leone, and G. Terracina, editors, *Proc. 8th Intl. Conf. Logic Programming and Nonmonotonic Reasoning*, pages 211–226, Diamante, 2005. Springer-Verlag.

[65] N. Kartha and V. Lifschitz. Actions with indirect effects (preliminary report). In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. 4th Intl. Conf. on Knowledge Representation and Reasoning (KR'94)*, pages 341–350, Bonn, 1994. Morgan Kaufmann Publishers.

[66] H. Katsuno and A. Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52(3):263–294, 1991.

[67] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In P. Gärdenfors, editor, *Belief revision*, pages 183–203. Cambridge University Press, 1992.

[68] R. Kowalski. Logic and modules, 2005. Available at `http://www.doc.ic.ac.uk/~rak`.

[69] M. Kracht and F. Wolter. Properties of independently axiomatizable bimodal logics. *J. of Symbolic Logic*, 56(4):1469–1485, 1991.

[70] M. Kracht and F. Wolter. Simulation and transfer results in modal logic: A survey. *Studia Logica*, 59:149–177, 1997.

[71] R. Kwok, N. Foo, and A. Nayak. Coherence of laws. In Sorge et al. [109], pages 1400–1401.

[72] J. Lang, F. Lin, and P. Marquis. Causal theories of action – a computational core. In Sorge et al. [109], pages 1073–1078.

[73] R. Li and L.M. Pereira. What is believed is what is explained. In Shrobe and Senator [107], pages 550–555.

[74] P. Liberatore. On the compilability of diagnosis, planning, reasoning about actions, belief revision, etc. In Cohn et al. [19], pages 144–155.

[75] P. Liberatore. A framework for belief update. In *Proc. 7th Eur. Conf. on Logics in Artificial Intelligence (JELIA'2000)*, pages 361–375, 2000.

[76] V. Lifschitz. Circumscription. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming – Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3, pages 297–352. Oxford University Press, 1994.

[77] V. Lifschitz and W. Ren. Towards a modular action description language. In *Proc. 21st Natl. Conf. on Artificial Intelligence (AAAI'2006)*, Boston, 2006. AAAI Press/MIT Press.

[78] F. Lin. Embracing causality in specifying the indirect effects of actions. In Mellish [92], pages 1985–1991.

[79] F. Lin. Embracing causality in specifying the indeterminate effects of actions. In Shrobe and Senator [107], pages 670–676.

[80] F. Lin and R. Reiter. State constraints revisited. *J. of Logic and Computation*, 4(5):655–678, 1994.

[81] P. Marquis. Knowledge compilation using theory prime implicates. In Mellish [92], pages 837–843.

[82] P. Marquis. Consequence finding algorithms. In D. Gabbay and Ph. Smets, editors, *Algorithms for Defeasible and Uncertain Reasoning, in S. Moral, J. Kohlas (Eds), Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 5, chapter 2, pages 41–145. Kluwer Academic Publishers, 2000.

[83] N. McCain and H. Turner. A causal theory of ramifications and qualifications. In Mellish [92], pages 1978–1984.

[84] N. McCain and H. Turner. Causal theories of action and change. In M. Witbrock and A. Hauptmann, editors, *Proc. 14th Natl. Conf. on Artificial Intelligence (AAAI'97)*, pages 460–465, Providence, 1997. AAAI Press/MIT Press.

[85] J. McCarthy. Epistemological problems of artificial intelligence. In N. Sridharan, editor, *Proc. 5th Intl. Joint Conf. on Artificial Intelligence (IJCAI'77)*, pages 1038–1044, Cambridge, MA, 1977. Morgan Kaufmann Publishers.

[86] J. McCarthy. Circumscription, a form of nonmonotonic reasoning. *Artificial Intelligence*, 13(1–2):27–39, 1980.

[87] J. McCarthy. Applications of circumscription to formalizing common-sense knowledge. *Artificial Intelligence*, 28(1):89–116, 1986.

[88] J. McCarthy. *Mathematical logic in artificial intelligence*. Daedalus, 1988.

[89] J. McCarthy. Elaboration tolerance. In *Proc. Common Sense'98*, London, 1998. Available at `http://www-formal.stanford.edu/jmc/elaboration.html`.

[90] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, 1969.

[91] S. McIlraith. Integrating actions and state constraints: A closed-form solution to the ramification problem (sometimes). *Artificial Intelligence*, 116(1–2):87–121, 2000.

[92] C. Mellish, editor. *Proc. 14th Intl. Joint Conf. on Artificial Intelligence (IJCAI'95)*, Montreal, 1995. Morgan Kaufmann Publishers.

[93] H. Ohlbach. Semantics based translation methods for modal logics. *J. of Logic and Computation*, 1(5):691–746, 1991.

[94] H. Ohlbach. Translation methods for non-classical logics – an overview. *J. of the IGPL*, 1(1):69–90, 1993.

[95] E. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In R. Brachman, H. Levesque, and R. Reiter, editors, *Proc. 2nd Intl. Conf. on Knowledge Representation and Reasoning (KR'89)*, pages 324–332, Toronto, 1989. Morgan Kaufmann Publishers.

[96] F. Pirri and R. Reiter. Some contributions to the metatheory of the situation calculus. *Journal of the ACM*, 46(3):325–361, 1999.

[97] S. Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

[98] R. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 1992.

[99] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, 1991.

[100] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

[101] E. Sandewall. Assessments of ramifications methods that use static domain constraints. In L. Aiello, J. Doyle, and S. Shapiro, editors, *Proc. 5th Intl. Conf. on Knowledge Representation and Reasoning (KR'96)*, pages 99–110, Cambridge, MA, 1996. Morgan Kaufmann Publishers.

[102] L. Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H. Kyberg, R. Loui, and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer Academic Publishers, 1990.

[103] L. Schubert. Explanation closure, action closure and the Sandewall test suite for reasoning about change. *J. of Logic and Computation*, 4(5):679–700, 1994.

[104] C. Schwind. Causality in action theories. *Linköping Electronic Articles in Computer and Information Science*, 4(4), 1999.

[105] M. Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT Press, Cambridge, MA, 1997.

[106] S. Shapiro, M. Pagnucco, Y. Lespérance, and H. Levesque. Iterated belief change in the situation calculus. In Cohn et al. [20], pages 527–538.

[107] H. Shrobe and T. Senator, editors. *Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96)*, Portland, 1996. AAAI Press/MIT Press.

[108] I. Sommerville. *Software Engineering*. Addison Wesley, 1985.

[109] V. Sorge, S. Colton, M. Fisher, and J. Gow, editors. *Proc. 18th Intl. Joint Conf. on Artificial Intelligence (IJCAI'03)*, Acapulco, 2003. Morgan Kaufmann Publishers.

[110] H. Stuckenschmidt and M. Klein. Integrity and change in modular ontologies. In Sorge et al. [109], pages 900–908.

[111] B. ten Cate, W. Conradie, M. Marx, and Y. Venema. Definitorially complete description logics. In Doherty et al. [29], pages 79–89.

[112] M. Thielscher. Computing ramifications by postprocessing. In Mellish [92], pages 1994–2000.

[113] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[114] J. Vennekens and M. Denecker. An algebraic account of modularity in ID-logic. In C. Baral, G. Greco, N. Leone, and G. Terracina, editors, *Proc. 8th Intl. Conf. Logic Programming and Nonmonotonic Reasoning*, pages 291–303, Diamante, 2005. Springer-Verlag.

[115] J. Vennekens, D. Gilis, and M. Denecker. Splitting an operator: An algebraic modularity result and its application to auto-epistemic logic. In *Workshop on Nonmonotonic Reasoning (NMR'04)*, Whistler, 2004.

[116] M.-A. Winslett. Reasoning about action using a possible models approach. In R. Smith and T. Mitchell, editors, *Proc. 7th Natl. Conf. on Artificial Intelligence (AAAI'88)*, pages 89–93, St. Paul, 1988. Morgan Kaufmann Publishers.

[117] M.-A. Winslett. Updating logical databases. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 133–174. Oxford University Press, 1995.

[118] D. Zhang, S. Chopra, and N. Foo. Consistency of action descriptions. In M. Ishizuka and A. Sattar, editors, *Proc. 7th Pacific Rim Intl. Conf. on Artificial*

Intelligence: Trends in Artificial Intelligence, number 2417 in LNCS, pages 70–79. Springer-Verlag, 2002.

[119] D. Zhang and N. Foo. EPDL: A logic for causal reasoning. In B. Nebel, editor, *Proc. 17th Intl. Joint Conf. on Artificial Intelligence (IJCAI'01)*, pages 131–138, Seattle, 2001. Morgan Kaufmann Publishers.

[120] D. Zhang and N. Foo. Interpolation properties of action logic: Lazy-formalization to the frame problem. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Proc. 8th Eur. Conf. on Logics in Artificial Intelligence (JELIA'2002)*, number 2424 in LNCS, pages 357–368. Springer-Verlag, 2002.

[121] D. Zhang and N. Foo. Frame problem in dynamic logic. *J. of Applied Non-Classical Logics (JANCL)*, 15(2):215–239, 2005.

Long Proofs of Chapter 4

Proof of Theorem 4.1

Let the underlying logic be a fusion, and let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$ be such that $\mathcal{T}$ is partitioned. If $\mathcal{D}$ is propositionally modular, then $\mathcal{D}$ is modular.

Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$ be propositionally modular. Suppose that for some Φ $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T}^{\emptyset} \cup \mathcal{T}^{act(\Phi)} \rangle \not\models \Phi$, i.e., $\mathcal{T}^{\emptyset} \cup \mathcal{T}^{act(\Phi)} \not\models_{\mathsf{PDL}} \Phi$. Hence there is a PDL-model $\mathscr{M} = \langle W, R \rangle$ such that $\models^{\mathscr{M}} \mathcal{T}^{act(\Phi)} \wedge \mathcal{T}^{\emptyset}$, and $\not\models^{\mathscr{M}} \Phi$. This means that there is some $w \in W$ such that $\not\models^{\mathscr{M}}_{w} \Phi$. We prove that $\mathcal{D} \not\models \Phi$ by constructing from $\mathscr{M}$ a model $\mathscr{M}'$ such that $\models^{\mathscr{M}'} \mathcal{T}$ and $\not\models^{\mathscr{M}'}_{w} \Phi$.

First, as our logic is an extension of classical propositional logic and it is compact, propositional modularity implies that for every propositional valuation $val \subseteq 2^{\mathfrak{Lit}}$ which is a model of $\mathcal{T}^{\emptyset}$, there is a possible worlds model $\mathscr{M}_{val} = \langle W_{val}, R_{val} \rangle$ such that $\models^{\mathscr{M}_{val}} \mathcal{T}$, and $val \in W_{val}$, i.e., for every propositional valuation of $\mathcal{T}^{\emptyset}$, there is a model of $\mathcal{T}$ containing that valuation.

Second, taking the disjoint union of all these models, we obtain a model $\mathscr{M}' = \langle W', R' \rangle$ such that $\models^{\mathscr{M}'} \mathcal{T}$, and for every propositional valuation $val \subseteq 2^{\mathfrak{Lit}}$ of $\mathcal{T}^{\emptyset}$, there is a possible world $w' \in W'$ such that $w' = val$.

Now, we can use $\mathscr{M}'$ to adjust those accessibility relations R_a of $\mathscr{M}$ whose a does not appear in Φ, in a way such that the resulting model satisfies the rest of the theory $\mathcal{T} \setminus \mathcal{T}^{act(\Phi)}$. Let $\mathscr{M}'' = \langle W'', R'' \rangle$ be such that

- $W'' = \{u_v : u \in W, v \in W', \text{and } u = v\}$;
- if $a \in act(\Phi)$, then $u_v R''_a u'_{v'}$ if and only if $u R_a u'$;
- if $a \notin act(\Phi)$, then $u_v R''_a u'_{v'}$ if and only if $v R_a v'$; and
- $u_v = u = v$.

We have $W'' \neq \emptyset$ because $\models^{\mathscr{M}} \mathcal{T}^{\emptyset}$. $\mathscr{M}''$ is a model of the underlying logic because the latter is a fusion. Then, for the sublanguage constructed from $act(\Phi)$, it can be proved by structural induction that for every formula Φ' of the sublanguage and every $u \in W$ and $v \in W'$, $\models^{\mathscr{M}}_{u} \Phi'$ if and only if $\models^{\mathscr{M}''}_{u_v} \Phi'$. The same can be proved for the sublanguage constructed from $\mathfrak{Act} \setminus act(\Phi)$. As, by hypothesis, $\mathcal{T}$ is partitioned, $\mathcal{T}^{\emptyset}$ and each $\mathcal{T}^{a}$ are in at least one of these sublanguages, thus we have proved that $\models^{\mathscr{M}''} \mathcal{T}$, and $\not\models^{\mathscr{M}''}_{w_v} \Phi$ for every v. Hence $\mathcal{D} \not\models \Phi$. ∎

Proof of Theorem 4.4

An action theory $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{T} \rangle$ is modular if and only if $\mathcal{S}_{imp} = \emptyset$.

($\Rightarrow$): Suppose $\mathcal{S}_{imp} = \emptyset$. Therefore, for all subsets $\{\varphi_1 \to [a]\psi_1, \ldots, \varphi_n \to [a]\psi_n\}$ of $\mathcal{T}^{a}$ and all $\varphi' \to \langle a \rangle \top \in \mathcal{T}^{a}$, we have that

$$\text{if } \mathcal{T}^{\emptyset} \cup \{\varphi', \varphi_1, \ldots, \varphi_n\} \not\models_{\mathsf{CPL}} \bot, \text{ then } \mathcal{T}^{\emptyset} \cup \{\psi_1, \ldots, \psi_n\} \not\models_{\mathsf{CPL}} \bot. \tag{A.1}$$

By Theorem 4.1, it suffices to prove that $\mathcal{D}$ is propositionally modular. Therefore, suppose $\mathcal{T}^{\emptyset} \not\models_{\mathsf{CPL}} \varphi$ for some propositional φ. Let W be the set of all propositional valuations satisfying $\mathcal{T}^{\emptyset}$ that falsify φ. As $\mathcal{T}^{\emptyset} \not\models_{\mathsf{CPL}} \varphi$, $\mathcal{T}^{\emptyset} \cup \{\neg\varphi\}$ is satisfiable, hence W must be nonempty. For every $w \in W$ let

$$\mathcal{E}^{a}_{\varphi}(w) = \{\varphi_i \,:\, \varphi_i \to [a]\psi_i \in \mathcal{T}^{a} \text{ and } w \text{ satisfies } \varphi_i\}$$

$$\mathcal{X}^{a}_{\varphi}(w) = \{\varphi_i \,:\, \varphi_i \to \langle a \rangle \top \in \mathcal{X}^{a} \text{ and } w \text{ satisfies } \varphi_i\}$$

We define R_a such that wR_aw' if and only if

- $\mathcal{X}^{a}_{\varphi}(w) \neq \emptyset$; and
- w' satisfies ψ_i for every $\varphi_i \to [a]\psi_i \in \mathcal{T}^{a}$ such that $\varphi_i \in \mathcal{E}^{a}_{\varphi}(w)$.

We then obtain a model $\mathscr{M} = \langle W, R \rangle$. We have that $\models^{\mathscr{M}} \mathcal{T}^{\emptyset}$, by the definition of W. Moreover, for every $w \in W$ and every $\varphi_i \to [a]\psi_i \in \mathcal{T}^{a}$, if $\models^{\mathscr{M}}_{w} \varphi_i$, then, by the definition of R_a, $\models^{\mathscr{M}}_{w'} \psi_i$ for all $w' \in W$ such that wR_aw'. We also have that for every $w \in W$ and every $\varphi_i \to \langle a \rangle \top \in \mathcal{T}^{a}$, if $\models^{\mathscr{M}}_{w} \varphi_i$, then from (A.1) and the definition of R_a, there exists at least one w' such that wR_aw'.

Hence, $\models^{\mathscr{M}} \mathcal{T}$. Clearly $\not\models^{\mathscr{M}} \varphi$, by the definition of W. Thus we have $\mathcal{T} \not\models_{\mathsf{PDL}} \varphi$, and then $\mathcal{D} \not\models \varphi$.

($\Leftarrow$): Straightforward, by the soundness result (Theorem 4.3). ∎

Long Proofs of Chapter 5

Proof of Theorem 5.1

Let $\mathcal{T}$ be the set of global axioms (5.4)–(5.8). Then

$$\begin{aligned}\mathcal{T} \models_{\mathsf{DPDL}^+} \quad & (\forall a.(Poss(a) \rightarrow ([a]p \leftrightarrow \\ & ((a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ & (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p))))))) \\ & \leftrightarrow \\ & (\forall a.([a]p \leftrightarrow \\ & (\neg Poss(a) \vee \\ & (a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ & (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p))))))\end{aligned}$$

Let φ denote the formula

$$\begin{aligned}& (a = a_1 \wedge Cond^+(a_1, p)) \vee \ldots \vee (a = a_n \wedge Cond^+(a_n, p)) \vee \\ & (p \wedge \neg(a = a'_1 \wedge Cond^-(a'_1, p)) \wedge \ldots \wedge \neg(a = a'_m \wedge Cond^-(a'_m, p)))\end{aligned}$$

($\rightarrow$): We are going to show that

$$\mathcal{T} \models_{\mathsf{DPDL}^+} (\forall a.(Poss(a) \rightarrow ([a]p \leftrightarrow \varphi))) \rightarrow (\forall a.([a]p \leftrightarrow (\neg Poss(a) \vee \varphi)))$$

1. $Poss(a) \rightarrow ([a]p \leftrightarrow \varphi)$, from hypothesis
2. $Poss(a) \rightarrow ([a]p \rightarrow \varphi)$, from 1. by classical logic
3. $(Poss(a) \wedge [a]p) \rightarrow \varphi$, from 2. by classical logic
4. $([a]p \wedge Poss(a)) \rightarrow \varphi$, from 3. by classical logic

5. $[a]p \rightarrow (Poss(a) \rightarrow \varphi)$, from 4. by classical logic
6. $Poss(a) \leftrightarrow \neg[a]\bot$, from global axiom (5.4)
7. $\neg Poss(a) \rightarrow [a]\bot$, from 6. and classical logic
8. $[a](\bot \rightarrow p)$, RN on $\bot \rightarrow p$
9. $[a]\bot \rightarrow [a]p$, from K on 8. and modus ponens
10. $\neg Poss(a) \rightarrow [a]p$, from 7. and 9. by classical logic
11. $Poss(a) \rightarrow (\varphi \rightarrow [a]p)$, from 1. by classical logic
12. $(\neg Poss(a) \wedge \varphi) \rightarrow [a]p$, from 10. by classical logic
13. $\neg Poss(a) \rightarrow (\varphi \rightarrow [a]p)$, from 12. by classical logic
14. $(Poss(a) \vee \neg Poss(a)) \rightarrow (\varphi \rightarrow [a]p)$, from 11. and 13. by classical logic
15. $\top \rightarrow (\varphi \rightarrow [a]p)$, from 14. by classical logic
16. $\varphi \rightarrow [a]p$, from 15. by classical logic
17. $(\neg Poss(a) \vee \varphi) \rightarrow [a]p$, from 10. and 16. by classical logic
18. $(Poss(a) \rightarrow \varphi) \rightarrow [a]p$, from 17. by classical logic
19. $[a]p \leftrightarrow (Poss(a) \rightarrow \varphi)$, from 5. and 18. by classical logic
20. $[a]p \leftrightarrow (\neg Poss(a) \vee \varphi)$, from 19. by classical logic

($\leftarrow$): We now prove that

$$\mathcal{T} \models_{\mathsf{DPDL}^+} (\forall a.([a]p \leftrightarrow (\neg Poss(a) \vee \varphi))) \rightarrow (\forall a.(Poss(a) \rightarrow ([a]p \leftrightarrow \varphi)))$$

1. $[a]p \leftrightarrow (\neg Poss(a) \vee \varphi)$, from hypothesis
2. $(\neg Poss(a) \vee \varphi) \rightarrow [a]p$, from 1. by classical logic
3. $(Poss(a) \rightarrow \varphi) \rightarrow [a]p$, from 2. by classical logic
4. $Poss(a) \rightarrow (\varphi \rightarrow [a]p)$, from 3. by classical logic
5. $[a]p \rightarrow (\neg Poss(a) \vee \varphi)$, from 1. by classical logic

6. $[a]p \rightarrow (Poss(a) \rightarrow \varphi)$, from 5. by classical logic
7. $([a]p \wedge Poss(a)) \rightarrow \varphi$, from 6. by classical logic
8. $Poss(a) \rightarrow ([a]p \rightarrow \varphi)$, from 7. by classical logic
9. $Poss(a) \rightarrow ([a]p \leftrightarrow \varphi)$, from 4. and 8. by classical logic

■

Proof of Theorem 5.3

Let the underlying logic be deterministic PDL, $\rightsquigarrow$ *be a dependence relation obtained from sets* $causes^{+}(.)$ *and* $causes^{-}(.)$, *and let* $\mathcal{T}$ *be the set of global axioms (5.4)–(5.8). Then*

(1) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee p$, *if* $a \not\rightsquigarrow p$ *and* $a \not\rightsquigarrow \neg p$;

(2) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee (p \wedge \neg Cond^{-}(a,p))$, *if* $a \not\rightsquigarrow p$ *and* $a \rightsquigarrow \neg p$;

(3) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee Cond^{+}(a,p) \vee p$, *if* $a \rightsquigarrow p$ *and* $a \not\rightsquigarrow \neg p$; *and*

(4) $\mathcal{T} \models_{\rightsquigarrow} [a]p \leftrightarrow \neg Poss(a) \vee Cond^{+}(a,p) \vee (p \wedge \neg Cond^{-}(a,p))$, *if* $a \rightsquigarrow p$ *and* $a \rightsquigarrow \neg p$.

Proving (1):
($\rightarrow$): We are about to prove $([a]p \wedge \neg p) \rightarrow \neg Poss(a)$.

1. $\neg p \rightarrow [a]\neg p$, from the hypothesis $a \not\rightsquigarrow p$
2. $([a]p \wedge \neg p) \rightarrow ([a]p \wedge [a]\neg p)$, from 1. by classical logic
3. $([a]p \wedge [a]\neg p) \rightarrow [a](p \wedge \neg p)$, by K and classical logic
4. $([a]p \wedge [a]\neg p) \rightarrow [a]\bot$, from 3. and classical logic
5. $([a]p \wedge \neg p) \rightarrow [a]\bot$, from 2. and 4. by classical logic
6. $[a]\bot \rightarrow \neg Poss(a)$, from global axiom (5.4)
7. $([a]p \wedge \neg p) \rightarrow \neg Poss(a)$, from 5. and 6. by classical logic

($\leftarrow$): We now prove $\neg Poss(a) \vee p \rightarrow [a]p$.

1. $p \rightarrow [a]p$, from the hypothesis $a \not\rightsquigarrow \neg p$
2. $\neg Poss(a) \rightarrow [a]\bot$, from global axiom (5.4)

3. $[a](\bot \to p)$, RN on $\bot \to p$
4. $[a]\bot \to [a]p$, from K on 3. and modus ponens
5. $\neg Poss(a) \to [a]p$, from 2. and 4. by classical logic
6. $\neg Poss(a) \vee p \to [a]p$, from 1. and 5. by classical logic

Proving (2):
($\to$): Let's show $([a]p \wedge \neg p) \to \neg Poss(a)$ and $([a]p \wedge Cond^{-}(a,p)) \to \neg Poss(a)$.

1. $\neg p \to [a]\neg p$, from the hypothesis $a \not\rightsquigarrow p$
2. $([a]p \wedge \neg p) \to ([a]p \wedge [a]\neg p)$, from 1. by classical logic
3. $([a]p \wedge [a]\neg p) \to [a](p \wedge \neg p)$, by K and classical logic
4. $([a]p \wedge [a]\neg p) \to [a]\bot$, from 3. and classical logic
5. $[a]\bot \to \neg Poss(a)$, from global axiom (5.4)
6. $([a]p \wedge [a]\neg p) \to \neg Poss(a)$, from 4. and 5. by classical logic
7. $([a]p \wedge \neg p) \to \neg Poss(a)$, from 2. and 6. by classical logic
8. $Cond^{-}(a,p) \to [a]\neg p$, by global axiom (5.7)
9. $([a]p \wedge Cond^{-}(a,p)) \to ([a]p \wedge [a]\neg p)$, from 8. by classical logic
10. $([a]p \wedge Cond^{-}(a,p)) \to [a]\bot$, from 9. and 4. by classical logic
11. $([a]p \wedge Cond^{-}(a,p) \to \neg Poss(a)$, from 10. and 5. by classical logic

($\leftarrow$): We are going to prove $\neg Poss(a) \vee (p \wedge \neg Cond^{-}(a,p)) \to [a]p$.

1. $\neg Poss(a) \to [a]\bot$, from global axiom (5.4)
2. $[a](\bot \to p)$, RN on $\bot \to p$
3. $[a]\bot \to [a]p$, from K on 2. and modus ponens
4. $\neg Poss(a) \to [a]p$, from 1. and 3. by classical logic
5. $(p \wedge \neg Cond^{-}(a,p)) \to [a]p$, from global axiom (5.8)
6. $\neg Poss(a) \vee (p \wedge \neg Cond^{-}(a,p)) \to [a]p$, from 4. and 5. by classical logic

Proving (3):
($\rightarrow$): We will prove $([a]p \wedge \neg Cond^{+}(a,p) \wedge \neg p) \rightarrow \neg Poss(a)$.

1. $(\neg Cond^{+}(a,p) \wedge \neg p) \rightarrow [a]\neg p$, by global axiom (5.6)
2. $([a]p \wedge \neg Cond^{+}(a,p) \wedge \neg p) \rightarrow ([a]p \wedge [a]\neg p)$, from 1. by classical logic
3. $([a]p \wedge [a]\neg p) \rightarrow [a](p \wedge \neg p)$, by K and classical logic
4. $([a]p \wedge [a]\neg p) \rightarrow [a]\bot$, from 3. and classical logic
5. $([a]p \wedge \neg Cond^{+}(a,p) \wedge \neg p) \rightarrow [a]\bot$, from 2. and 4. by classical logic
6. $[a]\bot \rightarrow \neg Poss(a)$, from global axiom (5.4)
7. $([a]p \wedge \neg Cond^{+}(a,p) \wedge \neg p) \rightarrow \neg Poss(a)$, from 5. and 6. by classical logic

($\leftarrow$): We are about to prove $\neg Poss(a) \vee Cond^{+}(a,p) \vee p \rightarrow [a]p$

1. $\neg Poss(a) \rightarrow [a]\bot$, from global axiom (5.4)
2. $[a](\bot \rightarrow p)$, RN on $\bot \rightarrow p$
3. $[a]\bot \rightarrow [a]p$, from K on 2. and modus ponens
4. $\neg Poss(a) \rightarrow [a]p$, from 1. and 3. by classical logic
5. $p \rightarrow [a]p$, by hypothesis $a \not\rightsquigarrow \neg p$
6. $Cond^{+}(a,p) \rightarrow [a]p$, from global axiom (5.5)
7. $\neg Poss(a) \vee Cond^{+}(a,p) \vee p \rightarrow [a]p$, from 4., 5. and 6. by classical logic

Proving (4):
($\rightarrow$): We prove $([a]p \wedge \neg Cond^{+}(a,p) \wedge \neg(p \wedge \neg Cond^{-}(a,p))) \rightarrow \neg Poss(a)$

1. $(\neg Cond^{+}(a,p) \wedge \neg p) \rightarrow [a]\neg p$, from global axiom (5.6)
2. $([a]p \wedge \neg Cond^{+}(a,p) \wedge \neg p) \rightarrow ([a]p \wedge [a]\neg p)$, from 1. by classical logic
3. $Cond^{-}(a,p) \rightarrow [a]\neg p$, by global axiom (5.7)
4. $([a]p \wedge \neg Cond^{+}(a,p) \wedge Cond^{-}(a,p)) \rightarrow ([a]p \wedge \neg Cond^{+}(a,p) \wedge [a]\neg p)$, from 3. by classical logic
5. $([a]p \wedge \neg Cond^{+}(a,p) \wedge [a]\neg p) \rightarrow ([a]p \wedge [a]\neg p)$, by classical logic

6. $([a]p \wedge \neg Cond^+(a,p) \wedge Cond^-(a,p)) \rightarrow ([a]p \wedge [a]\neg p)$, from 4. and 5. by classical logic
7. $[a]p \wedge \neg Cond^+(a,p) \wedge \neg p \vee [a]p \wedge \neg Cond^+(a,p) \wedge Cond^-(a,p) \rightarrow [a]p \wedge [a]\neg p$, from 2. and 4. by classical logic
8. $([a]p \wedge \neg Cond^+(a,p) \wedge \neg(p \wedge \neg Cond^-(a,p))) \rightarrow ([a]p \wedge [a]\neg p)$, from 7. by classical logic
9. $([a]p \wedge [a]\neg p) \rightarrow [a](p \wedge \neg p)$, by K and classical logic
10. $([a]p \wedge [a]\neg p) \rightarrow [a]\bot$, from 9. and classical logic
11. $([a]p \wedge \neg Cond^+(a,p) \wedge \neg(p \wedge \neg Cond^-(a,p))) \rightarrow [a]\bot$, from 8. and 10. by classical logic
12. $[a]\bot \rightarrow \neg Poss(a)$, from global axiom (5.4)
13. $([a]p \wedge \neg Cond^+(a,p) \wedge \neg(p \wedge \neg Cond^-(a,p))) \rightarrow \neg Poss(a)$, from 11. and 12. by classical logic

($\leftarrow$): We will prove $\neg Poss(a) \vee Cond^+(a,p) \vee (p \wedge \neg Cond^-(a,p)) \rightarrow [a]p$

1. $\neg Poss(a) \rightarrow [a]\bot$, from global axiom (5.4)
2. $[a](\bot \rightarrow p)$, RN on $\bot \rightarrow p$
3. $[a]\bot \rightarrow [a]p$, from K on 2. and modus ponens
4. $\neg Poss(a) \rightarrow [a]p$, from 1. and 3. by classical logic
5. $Cond^+(a,p) \rightarrow [a]p$, from global axiom (5.5)
6. $(p \wedge \neg Cond^-(a,p)) \rightarrow [a]p$, by global axiom (5.8)
7. $\neg Poss(a) \vee Cond^+(a,p) \vee (p \wedge \neg Cond^-(a,p)) \rightarrow [a]p$, from 4., 5. and 6. by classical logic

■

Long Proofs of Chapter 7

We recall that $\models_{\mathsf{CPL}}$ is logical consequence in classical propositional logic, and $PI(\mathcal{T}^{\emptyset})$ is the set of prime implicates of the set $\mathcal{T}^{\emptyset}$ of classical formulas.

Before giving the proof of the theorems, we recall some properties of prime implicates [81, 82] and of the function *NewCons*(.) [61] (see Section 7.3). Let $\varphi \in \mathfrak{Fml}$, $\mathcal{T}^{\emptyset} \subseteq \mathfrak{Fml}$ finite (identified with the conjunction of its formulas), and χ be a clause. Then

1. $\models_{\mathsf{CPL}} \varphi \leftrightarrow \bigwedge PI(\varphi)$ [82, Corollary 3.2].
2. $PI(\mathcal{T}^{\emptyset}) \cup NewCons(\varphi, \mathcal{T}^{\emptyset}) = PI(\mathcal{T}^{\emptyset} \wedge \varphi)$ (by definition of *NewCons*(.)).
3. $\models_{\mathsf{CPL}} (\mathcal{T}^{\emptyset} \wedge \varphi) \leftrightarrow (\mathcal{T}^{\emptyset} \wedge NewCons(\varphi, \mathcal{T}^{\emptyset}))$ (from 1 and 2)
4. If $PI(\varphi) \models_{\mathsf{CPL}} \chi$, then there is $\chi' \in PI(\varphi)$ such that $\chi' \models_{\mathsf{CPL}} \chi$ [82, Proposition 3.4].

Proof of Theorem 7.3

Let $\mathcal{S}_{imp^*}$ be the output of Algorithm 7.1 on input $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$. Then $\mathcal{D}^a$ satisfies Postulate **PS** *if and only if $\mathcal{S}_{imp^*} = \emptyset$.*

Let $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$ be an action theory for a, and let $\varphi \rightarrow \langle a \rangle \top \in \mathcal{X}^a$, $\mathcal{C}^a = \mathcal{E}^a \cup \mathcal{I}^a$, and $\widehat{\mathcal{C}^a} \subseteq \mathcal{C}^a$. We define:

$$\varphi_{\widehat{\mathcal{C}^a}} = \bigwedge \{ \varphi_i \,:\, \varphi_i \rightarrow [a]\psi_i \in \widehat{\mathcal{C}^a} \}$$

$$\psi_{\widehat{\mathcal{C}^a}} = \bigwedge \{ \psi_i \,:\, \varphi_i \rightarrow [a]\psi_i \in \widehat{\mathcal{C}^a} \}$$

Moreover, let $indep_a = \{ \neg \ell \,:\, a \not\rightsquigarrow \ell \}$.

Lemma C.1

Let $indep'_a \subseteq indep_a$. $\mathcal{S} \cup \{\psi_{\hat{\mathcal{C}}^a}\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$ *if and only if* $\mathcal{S} \cup NewCons(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$.

Proof:

$$\mathcal{S} \cup \{\psi_{\hat{\mathcal{C}}^a}\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$$

if and only if

$$PI(\mathcal{S} \cup \{\psi_{\hat{\mathcal{C}}^a}\}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot \text{ (by Property 1)}$$

if and only if

$$PI(\mathcal{S}) \cup NewCons(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot \text{ (by Property 2)}$$

if and only if

$$\mathcal{S} \cup NewCons(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot \text{ (by Property 1).}$$

■

Lemma C.2

Let $indep'_a \subseteq indep_a$. *If* $\mathcal{S} \cup NewCons(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$, *then there exists* $\chi \in NewCons(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S})$ *such that* $\mathcal{S} \cup \{\chi\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$.

Proof:

$$\mathcal{S} \cup NewCons(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$$

if and only if

$$PI(\mathcal{S}) \cup NewCons(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot \text{ (by Property 1)}$$

if and only if

$$PI(\mathcal{S} \cup \{\psi_{\hat{\mathcal{C}}^a}\}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot \text{ (by Property 2)}$$

if and only if

$$PI(\mathcal{S} \cup \{\psi_{\hat{\mathcal{C}}^a}\}) \models_{\overline{\mathsf{CPL}}} \neg \bigwedge \{\neg \ell_i : \neg \ell_i \in indep'_a\}$$

if and only if

$$PI(\mathcal{S} \cup \{\psi_{\hat{\mathcal{C}}^a}\}) \models_{\overline{\mathsf{CPL}}} \bigvee \{\ell_i : \neg \ell_i \in indep'_a\}$$

if and only if there exists $\chi \in PI(\mathcal{S} \cup \{\psi_{\hat{\mathcal{C}^a}}\})$ such that

$$\chi \models_{\overline{\mathsf{CPL}}} \bigvee\{\ell_i : \neg\ell_i \in indep'_a\} \text{ (by Property 4)}$$

if and only if

$$\{\chi\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$$

if and only if

$$\mathcal{S} \cup \{\chi\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot.$$

■

Lemma C.3

Let $indep'_a \subseteq indep_a$. If we have both $\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}^a}}\} \cup indep'_a \not\models_{\overline{\mathsf{CPL}}} \bot$ and $\mathcal{S} \cup NewCons(\psi_{\hat{\mathcal{C}^a}}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$, then there exists $\chi \in NewCons(\psi_{\hat{\mathcal{C}^a}}, \mathcal{S})$ such that $\mathcal{S} \cup \{\chi\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$.

Proof:

By Lemma C.2 and classical logic. ■

Lemma C.4

Let $indep'_a \subseteq indep_a$. If we have both $\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}^a}}\} \cup indep'_a \not\models_{\overline{\mathsf{CPL}}} \bot$ and $\mathcal{S} \cup NewCons(\psi_{\hat{\mathcal{C}^a}}, \mathcal{S}) \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$, then there exists $\chi \in NewCons(\psi_{\hat{\mathcal{C}^a}}, \mathcal{S})$ such that both $\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}^a}}\} \cup indep'_a \not\models_{\overline{\mathsf{CPL}}} \bot$ and $\mathcal{S} \cup \{\chi\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$.

Proof:

Trivially, by Lemma C.3. ■

Lemma C.5

Let $indep'_a \subseteq indep_a$. If $\chi \in NewCons(\psi_{\hat{\mathcal{C}^a}}, \mathcal{S})$ is such that $\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}^a}}\} \cup indep'_a \not\models_{\overline{\mathsf{CPL}}} \bot$ and $\mathcal{S} \cup \{\chi\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$, then both $\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}^a}}\} \cup \{\neg\ell_i : \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\overline{\mathsf{CPL}}} \bot$ and $\mathcal{S} \cup \{\chi\} \cup \{\neg\ell_i : \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \models_{\overline{\mathsf{CPL}}} \bot$.

Proof:

Let $\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}^a}}\} \cup indep'_a \not\models_{\overline{\mathsf{CPL}}} \bot$ and $\chi \in NewCons(\psi_{\hat{\mathcal{C}^a}}, \mathcal{S})$ be such that $\mathcal{S} \cup \{\chi\} \cup indep'_a \models_{\overline{\mathsf{CPL}}} \bot$.

If $\chi = \bot$, the result is trivial. Otherwise, we have the following cases:

- If $atm(\chi) \not\subset atm(indep'_a)$, then the premise is false (and the lemma trivially holds).
- If $atm(\chi) = atm(indep'_a)$, the lemma holds.

- Let $atm(\chi) \subset atm(indep'_a)$. Then, from

$$\mathcal{S} \cup \{\varphi, \varphi_{\mathcal{C}^a}\} \cup indep'_a \not\models_{\mathsf{CPL}} \bot \text{ (the hypothesis)}$$

it follows

$$\mathcal{S} \cup \{\varphi, \varphi_{\mathcal{C}^a}\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\mathsf{CPL}} \bot.$$

From

$$\mathcal{S} \cup \{\chi\} \cup indep'_a \models_{\mathsf{CPL}} \bot \text{ (hypothesis)}$$

and because

$$\mathcal{S} \cup indep'_a \not\models_{\mathsf{CPL}} \bot,$$

it follows

$$\mathcal{S} \cup \{\chi\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \models_{\mathsf{CPL}} \bot.$$

■

Lemma C.6

If $\chi \in NewCons(\psi_{\mathcal{C}^a}, \mathcal{S})$ *is such that both* $\mathcal{S} \cup \{\varphi, \varphi_{\mathcal{C}^a}\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi$ *and* $a \not\rightsquigarrow \ell_i\} \not\models_{\mathsf{CPL}} \bot$ *and* $\mathcal{S} \cup \{\chi\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi$ *and* $a \not\rightsquigarrow \ell_i\} \models_{\mathsf{CPL}} \bot$*, then* $\mathcal{S} \cup \{\varphi, \varphi_{\mathcal{C}^a}, \neg\chi\} \not\models_{\mathsf{CPL}} \bot$ *and for all* $\ell_i \in \chi$, $a \not\rightsquigarrow \ell_i$.

Proof:

From

$$\mathcal{S} \cup \{\varphi, \varphi_{\mathcal{C}^a}\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\mathsf{CPL}} \bot$$

we conclude

$$\mathcal{S} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\mathsf{CPL}} \bot.$$

From this and the hypothesis

$$\mathcal{S} \cup \{\chi\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \models_{\mathsf{CPL}} \bot,$$

it follows

$$\mathcal{S} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \models_{\mathsf{CPL}} \neg\chi.$$

If $\mathcal{S} \models_{\mathsf{CPL}} \neg\chi$, then $\mathcal{S} \cup \{\psi_{\mathcal{C}^a}\} \models_{\mathsf{CPL}} \neg\chi$, and because $\chi \in NewCons(\psi_{\mathcal{C}^a}, \mathcal{S})$, we have $\chi \models_{\mathsf{CPL}} \neg\chi$, a contradiction. Hence $\mathcal{S} \cup \{\chi\} \not\models_{\mathsf{CPL}} \bot$.

Suppose now that there is a literal $\ell \in \chi$ such that $\neg\ell \notin \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\}$.

Then, the propositional valuation in which $\chi_{\ell \leftarrow true}$ satisfies

$$\mathcal{S} \cup \{\chi\} \cup \{\neg \ell_i \ : \ \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\},$$

and then

$$\mathcal{S} \cup \{\chi\} \cup \{\neg \ell_i \ : \ \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\overline{\mathsf{CPL}}} \bot.$$

Hence there cannot be such a literal, and then for all $\ell_i \in \chi$, $a \not\rightsquigarrow \ell_i$.

Now, from $a \not\rightsquigarrow \ell_i$ for all $\ell_i \in \chi$, we have $\models_{\overline{\mathsf{CPL}}} \bigwedge\{\neg \ell_i \ : \ \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \leftrightarrow \neg\chi$. From this and the hypothesis

$$\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}}^a}\} \cup \{\neg \ell_i \ : \ \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\overline{\mathsf{CPL}}} \bot$$

it follows $\mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}}^a}, \neg\chi\} \not\models_{\overline{\mathsf{CPL}}} \bot$. ■

Proof of Theorem 7.3

We are about to prove that $\mathcal{D}^a$ satisfies Postulate **PS** if and only if $\mathcal{S}_{imp^*} = \emptyset$.

($\Rightarrow$): Suppose $\mathcal{S}_{imp^*} \neq \emptyset$. Then at the first step of the algorithm there has been some $\varphi \rightarrow \langle a \rangle \top \in \mathcal{X}^a$ and some $\hat{\mathcal{C}}^a \subseteq \mathcal{C}^a$ such that for some $\chi \in \mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S})$, $\mathcal{D}^a \models \neg(\varphi \wedge \varphi_{\hat{\mathcal{C}}^a} \wedge \neg\chi)$ and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\overline{\mathsf{CPL}}}, \mathcal{S} \rangle \not\models \neg(\varphi \wedge \varphi_{\hat{\mathcal{C}}^a} \wedge \neg\chi)$. Hence $\mathcal{D}^a$ does not satisfy Postulate **PS**.

($\Leftarrow$): Suppose that $\mathcal{S}_{imp^*} = \emptyset$. Therefore for all $\varphi' \rightarrow \langle a \rangle \top \in \mathcal{X}^a$ and for all subsets $\hat{\mathcal{C}}^a \subseteq \mathcal{C}^a$, we have that

$$\begin{array}{c} \text{for all } \chi \in \mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}), \text{ if } \mathcal{S} \cup \{\varphi', \varphi_{\hat{\mathcal{C}}^a}, \neg\chi\} \not\models_{\overline{\mathsf{CPL}}} \bot, \\ \text{then there exists } \ell_i \in \chi \text{ such that } a \rightsquigarrow \ell_i \end{array} \tag{C.1}$$

From (C.1) and Lemma C.6, we get

$$\begin{array}{c} \text{for all } \chi \in \mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}), \text{ if } \mathcal{S} \cup \{\varphi, \varphi_{\hat{\mathcal{C}}^a}\} \cup \{\neg \ell_i \ : \ \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\overline{\mathsf{CPL}}} \bot, \\ \text{then } \mathcal{S} \cup \{\chi\} \cup \{\neg \ell_i \ : \ \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\overline{\mathsf{CPL}}} \bot \end{array}$$

From this and Lemma C.5, it follows that

$$\text{for all } \chi \in \mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}), \text{ if } \mathcal{S} \cup \{\varphi', \varphi_{\hat{\mathcal{C}}^a}\} \cup \mathit{indep}'_a \not\models_{\mathsf{CPL}} \bot, \text{ then } \mathcal{S} \cup \{\chi\} \cup \mathit{indep}'_a \not\models_{\mathsf{CPL}} \bot$$

This and Lemma C.4 gives us

$$\text{if } \mathcal{S} \cup \{\varphi', \varphi_{\hat{\mathcal{C}}^a}\} \cup \mathit{indep}'_a \not\models_{\mathsf{CPL}} \bot, \text{ then } \mathcal{S} \cup \mathit{NewCons}(\psi_{\hat{\mathcal{C}}^a}, \mathcal{S}) \cup \mathit{indep}'_a \not\models_{\mathsf{CPL}} \bot$$

From this and Lemma C.1, it follows that for all $\mathit{indep}'_a \subseteq \mathit{indep}_a$, for every $\varphi' \to \langle a \rangle \top \in \mathcal{X}^a$ and all $\hat{\mathcal{C}}^a \subseteq \mathcal{C}^a$,

$$\text{if } \mathcal{S} \cup \{\varphi', \varphi_{\hat{\mathcal{C}}^a}\} \cup \mathit{indep}'_a \not\models_{\mathsf{CPL}} \bot, \text{ then } \mathcal{S} \cup \{\psi_{\hat{\mathcal{C}}^a}\} \cup \mathit{indep}'_a \not\models_{\mathsf{CPL}} \bot. \tag{C.2}$$

Now, suppose $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \rangle \not\models \varphi$ for some propositional φ. We will build a model $\mathscr{M}$ such that $\mathscr{M}$ is a model for $\mathcal{D}^a$ that does not satisfy φ.

Let $\mathscr{M} = \langle W, R_a \rangle$ be such that $W = \mathit{valuations}(\mathcal{S})$, and R_a be such that for all $w, w' \in W$, wR_aw' if and only if

- $\models^{\mathscr{M}}_{w'} \psi_i$ for every $\varphi_i \to [a]\psi_i \in \mathcal{C}^a$ such that $\models^{\mathscr{M}}_{w} \varphi_i$; and
- $\models^{\mathscr{M}}_{w'} \neg\ell$ for all ℓ such that $a \not\rightsquigarrow \ell$ and $\models^{\mathscr{M}}_{w} \neg\ell$.

We have that $\mathscr{M}$ is a $\rightsquigarrow$-model, by the definition of R_a. By the definition of W, $\mathscr{M}$ is a model of $\mathcal{S}$. We have that $\mathscr{M}$ is a model of $\mathcal{E}^a$ and $\mathcal{I}^a$, too: for every $\varphi_i \to [a]\psi_i \in \mathcal{C}^a$ and every world $w \in W$, if $\models^{\mathscr{M}}_{w} \varphi_i$, then, by the definition of R_a, $\models^{\mathscr{M}}_{w'} \psi_i$ for all $w' \in W$ such that wR_aw'. Moreover, $\mathscr{M}$ is also a model of $\mathcal{X}^a$: for every $\varphi_i \to \langle a \rangle \top \in \mathcal{X}^a$ and every world $w \in W$, if $\models^{\mathscr{M}}_{w} \varphi_i$, then

$$\mathcal{E}^a(w) = \{\varphi_i \to [a]\psi_i \in \mathcal{E}^a \ : \models^{\mathscr{M}}_{w} \varphi_i\}, \text{ and } \mathit{indep}_a(w) = \{\neg\ell \ : \ a \not\rightsquigarrow \ell \text{ and } \models^{\mathscr{M}}_{w} \neg\ell\}$$

are such that $\mathcal{S} \cup \{\varphi_i, \varphi_{\mathcal{E}^a(w)}\} \cup \mathit{indep}_a(w) \not\models_{\mathsf{CPL}} \bot$, where

$$\varphi_{\mathcal{E}^a(w)} = \bigwedge \{\varphi_i \ : \ \varphi_i \to [a]\psi_i \in \mathcal{E}^a(w)\}$$

From this and (C.2), we have $\mathcal{S} \cup \{\psi_{\mathcal{E}^a(w)}\} \cup \mathit{indep}_a(w) \not\models_{\mathsf{CPL}} \bot$, where

$$\psi_{\mathcal{E}^a(w)} = \bigwedge\{\psi_i \,:\, \varphi_i \to [a]\psi_i \in \mathcal{E}^a(w)\}$$

As W is maximal, there exists w' such that $\not\models^{\mathscr{M}}_{w'} \psi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w)$. As R_a is maximal by definition, we have wR_aw'. Hence there exists at least one w' such that wR_aw', and $\models^{\mathscr{M}}_{w} \langle a\rangle\top$.

Hence, $\mathscr{M}$ is a model of $\mathcal{D}^a$. Clearly $\not\models^{\mathscr{M}} \varphi$, by the definition of W. Hence $\mathcal{D}^a \not\models \varphi$. Therefore $\mathcal{D}^a$ satisfies Postulate **PS**. ∎

Proof of Theorem 7.5

Let $\mathcal{I}^a_{imp}$ be the output of Algorithm 7.2 on input $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a\rangle$. If $\mathcal{D}^a$ satisfies Postulate **PS**, *then $\mathcal{D}^a$ satisfies Postulate* **PI** *if and only if $\mathcal{I}^a_{imp} = \emptyset$.*

Let $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a\rangle$ be an action theory for action a. For every $\hat{\mathcal{E}^a} \subseteq \mathcal{E}^a$ we define:

$$\varphi_{\hat{\mathcal{E}^a}} = \bigwedge\{\varphi_i \,:\, \varphi_i \to [a]\psi_i \in \hat{\mathcal{E}^a}\}$$

$$\psi_{\hat{\mathcal{E}^a}} = \bigwedge\{\psi_i \,:\, \varphi_i \to [a]\psi_i \in \hat{\mathcal{E}^a}\}$$

Moreover, let $\mathit{indep}_a = \{\neg\ell \,:\, a \not\rightsquigarrow \ell\}$.

Lemma C.7
If $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} (\varphi_{\hat{\mathcal{E}^a}} \wedge \mathit{indep}'_a) \to [a]\bot$ and $\mathcal{S} \cup \{\psi_{\hat{\mathcal{E}^a}}\} \cup \mathit{indep}'_a \models_{\mathsf{CPL}} \bot$, then there is $\chi \in \mathit{NewCons}(\psi_{\hat{\mathcal{E}^a}}, \mathcal{S})$ such that $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} (\varphi_{\hat{\mathcal{E}^a}} \wedge \neg\chi) \to [a]\bot$ and $a \not\rightsquigarrow \ell_i$ for all $\ell_i \in \chi$.

Proof:
Let $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} (\varphi_{\hat{\mathcal{E}^a}} \wedge \mathit{indep}'_a) \to [a]\bot$. Then there is a PDL-model $\mathscr{M} = \langle W, R_a\rangle$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{I}^a$ and $\not\models^{\mathscr{M}} (\varphi_{\hat{\mathcal{E}^a}} \wedge \mathit{indep}'_a) \to [a]\bot$. This means that there is a possible world $v \in W$ such that $\models^{\mathscr{M}}_{v} \varphi_{\hat{\mathcal{E}^a}} \wedge \mathit{indep}'_a$ and $\not\models^{\mathscr{M}}_{v} [a]\bot$. From $\models^{\mathscr{M}}_{v} \varphi_{\hat{\mathcal{E}^a}} \wedge \mathit{indep}'_a$, it follows

$$\mathcal{S} \cup \{\varphi_{\hat{\mathcal{E}^a}}\} \cup \mathit{indep}'_a \not\models_{\mathsf{CPL}} \bot \tag{C.3}$$

From hypothesis $\mathcal{S} \cup \{\psi_{\hat{\mathcal{E}^a}}\} \cup \mathit{indep}'_a \models_{\mathsf{CPL}} \bot$ and Lemma C.1, we get

$$\mathcal{S} \cup \mathit{NewCons}(\psi_{\hat{\mathcal{E}^a}}, \mathcal{S}) \cup \mathit{indep}'_a \models_{\mathsf{CPL}} \bot$$

and from this and Lemma C.2 we have that there is $\chi \in \mathit{NewCons}(\psi_{\widehat{\mathcal{E}^a}}, \mathcal{S})$ such that

$$\mathcal{S} \cup \{\chi\} \cup \mathit{indep}'_a \models_{\overline{\mathsf{CPL}}} \bot \tag{C.4}$$

From (C.3), (C.4) and classical logic, there is $\chi \in \mathit{NewCons}(\psi_{\widehat{\mathcal{E}^a}}, \mathcal{S})$ such that

$$\mathcal{S} \cup \{\varphi_{\widehat{\mathcal{E}^a}}\} \cup \mathit{indep}'_a \not\models_{\overline{\mathsf{CPL}}} \bot \text{ and } \mathcal{S} \cup \{\chi\} \cup \mathit{indep}'_a \models_{\overline{\mathsf{CPL}}} \bot$$

From this and Lemma C.5 it follows that there is $\chi \in \mathit{NewCons}(\psi_{\widehat{\mathcal{E}^a}}, \mathcal{S})$ such that

$$\mathcal{S} \cup \{\varphi, \varphi_{\widehat{\mathcal{E}^a}}\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \not\models_{\overline{\mathsf{CPL}}} \bot$$

and

$$\mathcal{S} \cup \{\chi\} \cup \{\neg \ell_i \,:\, \ell_i \in \chi \text{ and } a \not\rightsquigarrow \ell_i\} \models_{\overline{\mathsf{CPL}}} \bot$$

This and Lemma C.6 gives us that for all $\ell_i \in \chi$, $a \not\rightsquigarrow \ell_i$.

Now, because $\mathscr{M}$ above is such that $\models^{\mathscr{M}}_{\nu} \varphi_{\widehat{\mathcal{E}^a}} \wedge \mathit{indep}'_a$, from this and $\mathcal{S} \cup \{\chi\} \cup \mathit{indep}'_a \models_{\overline{\mathsf{CPL}}} \bot$, we have that $\models^{\mathscr{M}}_{\nu} \varphi_{\widehat{\mathcal{E}^a}} \wedge \neg\chi$. Because $\not\models^{\mathscr{M}}_{\nu} [a]\bot$, we therefore have $\mathcal{S}, \mathcal{I}^a \not\models_{\overline{\mathsf{PDL}}} (\varphi_{\widehat{\mathcal{E}^a}} \wedge \neg\chi) \rightarrow [a]\bot$. ∎

Proof of Theorem 7.5

We are about to prove that if $\mathcal{D}^a$ satisfies Postulate **PS**, then $\mathcal{D}^a$ satisfies Postulate **PI** if and only if $\mathcal{I}^a_{imp} = \emptyset$.

($\Rightarrow$): Straightforward, as every time $\mathcal{D}^a \models \varphi \rightarrow [a]\bot$, we have $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{I}^a \rangle \models \varphi \rightarrow [a]\bot$, and then $\mathcal{I}^a_{imp}$ never changes.

($\Leftarrow$): Suppose that $\mathcal{I}^a_{imp} = \emptyset$. Therefore for all subsets $\widehat{\mathcal{E}^a} \subseteq \mathcal{E}^a$, we have that

$$\begin{array}{c} \text{for all } \chi \in \mathit{NewCons}(\psi_{\widehat{\mathcal{E}^a}}, \mathcal{S}), \text{ if } \mathcal{S}, \mathcal{I}^a \not\models_{\overline{\mathsf{PDL}}} (\varphi_{\widehat{\mathcal{E}^a}} \wedge \neg\chi) \rightarrow [a]\bot, \\ \text{then there exists } \ell_i \in \chi \text{ such that } a \rightsquigarrow \ell_i \end{array} \tag{C.5}$$

From (C.5) and Lemma C.7, it follows that for all $\widehat{\mathcal{E}^a} \subseteq \mathcal{E}^a$,

$$\begin{array}{c} \text{if } \mathcal{S}, \mathcal{I}^a \not\models_{\overline{\mathsf{PDL}}} (\varphi_{\widehat{\mathcal{E}^a}} \wedge \mathit{indep}'_a) \rightarrow [a]\bot, \\ \text{then } \mathcal{S} \cup \{\psi_{\widehat{\mathcal{E}^a}}\} \cup \mathit{indep}'_a \not\models_{\overline{\mathsf{CPL}}} \bot. \end{array} \tag{C.6}$$

Suppose $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\overline{\mathsf{PDL}}}, \mathcal{S} \cup \mathcal{I}^a \rangle \not\models \varphi \rightarrow [a]\bot$ for some $\varphi \in \mathfrak{Fml}$. Then there exists a

PDL-model $\mathscr{M} = \langle W, R_a \rangle$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{I}^a$ and $\not\models^{\mathscr{M}} \varphi \rightarrow [a]\bot$. This means that there is a possible world $v \in W$ such that $\models^{\mathscr{M}}_{v} \varphi$ and $\not\models^{\mathscr{M}}_{v} [a]\bot$.

(We are going to build a model of $\mathcal{D}^a$, and hence conclude that $\mathcal{D}^a \not\models \varphi \rightarrow [a]\bot$.)

For given $w \in W$, we define:

$$\mathcal{I}^a(w) = \{\varphi_i \rightarrow [a]\bot \in \mathcal{I}^a \ : \models^{\mathscr{M}}_{w} \varphi_i\}$$

Because $\mathcal{D}^a$ satisfies Postulate **PS**, we can extend $\mathscr{M}$ to a big model $\mathscr{M}' = \langle W', R'_a \rangle$ such that $W = \mathit{valuations}(\mathcal{S})$, and R'_a is defined such that for all $w, w' \in W'$, $wR'_a w'$ if and only if

- $\models^{\mathscr{M}'}_{w'} \neg\ell$ for all ℓ such that $a \not\rightsquigarrow \ell$ and $\models^{\mathscr{M}'}_{w} \neg\ell$;
- $\models^{\mathscr{M}'}_{w'} \psi_i$ for every $\varphi_i \rightarrow [a]\psi_i \in \mathcal{E}^a$ such that $\models^{\mathscr{M}'}_{w} \varphi_i$; and
- $\mathcal{I}^a(w) = \emptyset$.

By definition, $\mathscr{M}'$ is a $\rightsquigarrow$-model. We also have $\models^{\mathscr{M}'} \mathcal{S}$, by the definition of W'. $\mathscr{M}'$ is a model of $\mathcal{E}^a$, too: for every $\varphi_i \rightarrow [a]\psi_i \in \mathcal{E}^a$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\models^{\mathscr{M}'}_{w'} \psi_i$ for all $w' \in W'$ such that $wR'_a w'$. Clearly $\mathscr{M}'$ is also a model of $\mathcal{I}^a$: for every $\varphi_i \rightarrow [a]\bot \in \mathcal{I}^a$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\mathcal{I}^a(w) \neq \emptyset$ and $R'_a(w) = \emptyset$. $\mathscr{M}'$ is a model of $\mathcal{X}^a$, too: for every $\varphi_i \rightarrow \langle a\rangle\top \in \mathcal{X}^a$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then

$$\mathcal{E}^a(w) = \{\varphi_i \rightarrow [a]\psi_i \in \mathcal{E}^a \ : \models^{\mathscr{M}}_{w} \varphi_i\}, \text{ and } \mathit{indep}_a(w) = \{\neg\ell \ : \ a \not\rightsquigarrow \ell \text{ and } \models^{\mathscr{M}}_{w} \neg\ell\}$$

are such that $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} (\varphi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w)) \rightarrow [a]\bot$, where

$$\varphi_{\mathcal{E}^a(w)} = \bigwedge\{\varphi_i \ : \ \varphi_i \rightarrow [a]\psi_i \in \mathcal{E}^a(w)\}$$

The justification is that $\mathcal{S}, \mathcal{I}^a \models_{\mathsf{PDL}} (\varphi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w)) \rightarrow [a]\bot$ would imply $\mathcal{D}^a \models (\varphi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w)) \rightarrow [a]\bot$, and as long as $\varphi_i \rightarrow \langle a\rangle\top \in \mathcal{X}^a$, $\mathcal{D}^a \models \neg(\varphi_i \wedge \varphi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w))$. As by hypothesis $\mathcal{D}^a$ satisfies **PS**, $\neg(\varphi_i \wedge \varphi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w)) \in \mathcal{S}$, and then $w \notin W'$.

Hence, from $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} (\varphi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w)) \rightarrow [a]\bot$ and (C.6), it follows that $\mathcal{S} \cup \{\psi_{\mathcal{E}^a(w)}\} \cup \mathit{indep}_a(w) \not\models_{\mathsf{CPL}} \bot$, where

$$\psi_{\mathcal{E}^a(w)} = \bigwedge\{\psi_i \ : \ \varphi_i \rightarrow [a]\psi_i \in \mathcal{E}^a(w)\}$$

As W' is maximal, there exists w' such that $\models^{\mathscr{M}'}_{w'} \psi_{\mathcal{E}^a(w)} \wedge \mathit{indep}_a(w)$. As R'_a is maximal

by definition, we have $wR'_a w'$. Hence there exists at least one w' such that $wR'_a w'$, and then $\models^{\mathscr{M}'}_{w'} \langle a \rangle \top$.

Therefore, $\mathscr{M}'$ is a model of $\mathcal{D}^a$.

Looking at $v \in W'$, we must have $\mathcal{S}, \mathcal{I}^a \not\vdash_{\mathsf{PDL}} (\varphi_{\mathcal{E}^a(v)} \wedge \mathit{indep}_a(v)) \rightarrow [a]\bot$, because otherwise $R_a(v) = \emptyset$, against the hypothesis that $\not\models^{\mathscr{M}}_{v} [a]\bot$. Hence, from (C.6) it follows that $\mathcal{S} \cup \{\psi_{\mathcal{E}^a(v)}\} \cup \mathit{indep}_a(v) \not\vdash_{\mathsf{CPL}} \bot$, and then there exists at least one v' such that $vR'_a v'$, and then $\models^{\mathscr{M}'}_{v'} \langle a \rangle \top$. From this it follows that $\mathcal{D}^a \not\models \varphi \rightarrow [a]\bot$. Therefore $\mathcal{D}^a$ satisfies Postulate **PI**. ■

Long Proofs of Chapter 8

Proof of Theorem 8.3

Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ *satisfy Postulate* **PS***. $\mathcal{D}$ *satisfies Postulate* **PI*** *if and only if* $\mathcal{D}^a = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle$ *satisfies Postulate* **PI** *for all* $a \in \mathfrak{Act}$.

($\Rightarrow$): Suppose that $\mathcal{D}^a \models \varphi \rightarrow [a]\bot$, i.e., $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi \rightarrow [a]\bot$. By monotonicity, $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \models_{\rightsquigarrow} \varphi \rightarrow [a]\bot$, too. Now suppose that $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle \not\models \varphi \rightarrow [a]\bot$, i.e., $\mathcal{S}, \mathcal{I}^a \not\models_{\mathsf{PDL}} \varphi \rightarrow [a]\bot$. Then there exists a possible worlds model $\mathscr{M} = \langle W, R_a \rangle$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{I}^a$ and there is a possible world $v \in W$ such that $\models^{\mathscr{M}}_{v} \varphi$ and $\not\models^{\mathscr{M}}_{v} [a]\bot$. Let $\mathscr{M}' = \langle W', R' \rangle$ be such that $W' = W$, and $R'_{a'} = \emptyset$, for all $a' \neq a$, and $R'_a = R_a$. Then $\models^{\mathscr{M}'} \mathcal{S} \wedge \mathcal{I}$, and then $\mathcal{S}, \mathcal{I} \not\models_{\mathsf{PDL}} \varphi \rightarrow [a]\bot$. Hence $\mathcal{D}$ does not satisfy **PI***.

($\Leftarrow$): Suppose that $\mathcal{D}$ does not satisfy Postulate **PI***. Then there exists $\varphi \in \mathfrak{Fml}$ such that $\mathcal{D} \models \varphi \rightarrow [a]\bot$ and $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I} \rangle \not\models \varphi \rightarrow [a]\bot$.

Claim: $\mathcal{D}^a \models \varphi \rightarrow [a]\bot$.

(Proof of the claim): Suppose $\mathcal{D}^a \not\models \varphi \rightarrow [a]\bot$, i.e., $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \not\models_{\rightsquigarrow} \varphi \rightarrow [a]\bot$. Then there exists a $\rightsquigarrow$-model $\mathscr{M} = \langle W, R_a \rangle$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{X}^a \wedge \mathcal{I}^a$ and $\not\models^{\mathscr{M}} \varphi \rightarrow [a]\bot$. This means that there is a possible world $v \in W$ such that $\models^{\mathscr{M}}_{v} \varphi$ and $\not\models^{\mathscr{M}}_{v} [a]\bot$, i.e., there is $v' \in W$ such that $R_a(v) = v'$.

(We extend $\mathscr{M}$ to all other actions $\mathcal{D}$ speaks of and obtain a model of $\mathcal{D}$.)

Given $w \in W$, for each $a_i \in \mathfrak{Act}$ we define:

$$\mathcal{I}^{a_i}(w) = \{\varphi_j \rightarrow [a_i]\bot \in \mathcal{I}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

$$\mathcal{X}^{a_i}(w) = \{\varphi_j \rightarrow \langle a_i \rangle \top \in \mathcal{X}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

Let $\mathscr{M}' = \langle W', R' \rangle$ be such that $W' = W$, and $R' = R_a \cup \bigcup_{a' \neq a} R_{a'}$, where for each $a' \neq a$ and every $w, w' \in W'$, $wR_{a'}w'$ if and only if

- $\models^{\mathscr{M}'}_{w'} \neg\ell$ for all ℓ such that $a' \not\rightsquigarrow \ell$ and $\models^{\mathscr{M}'}_{w} \neg\ell$.
- $\models^{\mathscr{M}'}_{w'} \psi_i$ for every $\varphi_i \to [a']\psi_i \in \mathcal{E}^{a'}$ such that $\models^{\mathscr{M}'}_{w} \varphi_i$; and
- $\mathcal{I}^{a'}(w) = \emptyset$;

By definition, $\mathscr{M}'$ is a model of the dependence relation $\rightsquigarrow$. Because, by hypothesis, $\mathcal{D}$ satisfies **PS***, there is no implicit static law, i.e., for every $a_i \in \mathfrak{Act}$ and every $w \in W'$, if $\mathcal{I}^{a_i}(w) \neq \emptyset$, then $\mathcal{X}^{a_i}(w) = \emptyset$. Then, as $W' = \mathit{valuations}(\mathcal{S})$, $\mathscr{M}'$ is a model of $\mathcal{S}$. We have that $\mathscr{M}'$ is a model of $\mathcal{E}$, too: it is a model of $\mathcal{E}^a$, and given $a' \neq a$, for every $\varphi_i \to [a']\psi_i \in \mathcal{E}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\models^{\mathscr{M}'}_{w'} \psi_i$ for all $w' \in W'$ such that $wR_{a'}w'$. Clearly $\mathscr{M}'$ is also a model of $\mathcal{I}$: it is a model of $\mathcal{I}^a$, and given $a' \neq a$, for every $\varphi_i \to [a']\bot \in \mathcal{I}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\mathcal{I}^{a'}(w) \neq \emptyset$ and $R_{a'}(w) = \emptyset$. $\mathscr{M}'$ is a model of $\mathcal{X}$, too: besides being a model of $\mathcal{X}^a$, for every $a' \neq a$ and all worlds $w \in W'$ such that $\mathcal{X}^{a'}(w) \neq \emptyset$ there is a world accessible by $R_{a'}$, because $R_{a'}(w) = \emptyset$ in this case would preclude $\mathcal{X}^{a'}(w) \neq \emptyset$, and otherwise $w \notin W'$, which is impossible as long as **PS*** is satisfied. Thus $\models^{\mathscr{M}'} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$, but if this is the case, $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \to [a]\bot$, hence we must have $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \models_{\rightsquigarrow} \varphi \to [a]\bot$, and then $\mathcal{D}^a \models \varphi \to [a]\bot$. (End of the proof of the claim.)

From $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I} \rangle \not\models \varphi \to [a]\bot$ it follows $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{I}^a \rangle \not\models \varphi \to [a]\bot$. Putting all the results together, we have that $\mathcal{D}^a$ does not satisfy Postulate **PI**. ■

Proof of Theorem 8.5

If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ *satisfies Postulate* **PS***, *then* $\mathcal{D} \models \varphi \to [a]\psi$ *if and only if* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle \models \varphi \to [a]\psi$.

($\Rightarrow$): Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ satisfy Postulate **PS***, and also suppose that $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle \not\models \varphi \to [a]\psi$, i.e., $\mathcal{S}, \mathcal{E}^a, \mathcal{I}^a \not\models_{\rightsquigarrow} \varphi \to [a]\psi$. Then there exists a $\rightsquigarrow$-model $\mathscr{M} = \langle W, R_a \rangle$, such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{I}^a$ and $\not\models^{\mathscr{M}} \varphi \to [a]\psi$. This means that there is a possible world $v \in W$ such that $\models^{\mathscr{M}}_{v} \varphi$ and $\not\models^{\mathscr{M}}_{v} [a]\psi$, i.e., there is $v' \in W$ such that $R_a(v) = v'$ and $\not\models^{\mathscr{M}}_{v'} \psi$.

(We will extend $\mathscr{M}$ to obtain a model of $\mathcal{D}$ and thus show that $\mathcal{D} \not\models \varphi \to [a]\psi$.)

Given $w \in W$, for each $a_i \in \mathfrak{Act}$ we define:

$$\mathcal{I}^{a_i}(w) = \{\varphi_j \to [a_i]\bot \in \mathcal{I}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

$$\mathcal{X}^{a_i}(w) = \{\varphi_j \to \langle a_i \rangle\top \in \mathcal{X}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

Let $\mathscr{M}' = \langle W', R' \rangle$ be such that $W' = W$, and $R' = R_a \cup \bigcup_{a' \neq a} R_{a'}$, where for each $a' \neq a$ and every $w, w' \in W'$, $wR_{a'}w'$ if and only if

- $\models^{\mathscr{M}'}_{w'} \neg\ell$ for all ℓ such that $a' \not\rightsquigarrow \ell$ and $\models^{\mathscr{M}'}_{w} \neg\ell$.
- $\models^{\mathscr{M}'}_{w'} \psi_i$ for every $\varphi_i \to [a']\psi_i \in \mathcal{E}^{a'}$ such that $\models^{\mathscr{M}'}_{w} \varphi_i$; and
- $\mathcal{I}^{a'}(w) = \emptyset$;

By definition, $\mathscr{M}'$ is a model of the dependence relation $\rightsquigarrow$. Because, by hypothesis, $\mathcal{D}$ satisfies **PS***, there is no implicit static law, i.e., for every $a_i \in \mathfrak{Act}$ and every $w \in W'$, if $\mathcal{I}^{a_i}(w) \neq \emptyset$, then $\mathcal{X}^{a_i}(w) = \emptyset$. Then, as $W' = \mathit{valuations}(\mathcal{S})$, $\mathscr{M}'$ is a model of $\mathcal{S}$. We have that $\mathscr{M}'$ is a model of $\mathcal{E}$, too: it is a model of $\mathcal{E}^a$, and given $a' \neq a$, for every $\varphi_i \to [a']\psi_i \in \mathcal{E}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\models^{\mathscr{M}'}_{w'} \psi_i$ for all $w' \in W'$ such that $wR_{a'}w'$. Clearly $\mathscr{M}'$ is also a model of $\mathcal{I}$: besides being a model of $\mathcal{I}^a$, given $a' \neq a$, for every $\varphi_i \to [a']\bot \in \mathcal{I}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\mathcal{I}^{a'}(w) \neq \emptyset$ and $R_{a'}(w) = \emptyset$. $\mathscr{M}'$ is a model of $\mathcal{X}$, too: it is a model of $\mathcal{X}^a$, and for every $a' \neq a$ and all worlds $w \in W'$ such that $\mathcal{X}^{a'}(w) \neq \emptyset$ there is a world accessible by $R_{a'}$, because $R_{a'}(w) = \emptyset$ in this case would preclude $\mathcal{X}^{a'}(w) \neq \emptyset$, and otherwise $w \notin W'$, which is impossible as long as **PS*** is satisfied. Thus $\models^{\mathscr{M}'} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$. Because there are $v, v' \in W'$ such that $\models^{\mathscr{M}'}_{v} \varphi$, $vR_a v'$ and $\not\models^{\mathscr{M}'}_{v'} \psi$, we have $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \to [a]\psi$, and then $\mathcal{D} \not\models \varphi \to [a]\psi$.

($\Leftarrow$): Suppose $\mathcal{D} \not\models \varphi \to [a]\psi$, i.e., $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \to [a]\psi$. Then there is a $\rightsquigarrow$-model $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$ and $\not\models^{\mathscr{M}} \varphi \to [a]\psi$. Then, given a, we have $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{X}^a \wedge \mathcal{I}^a$, and then $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{I}^a$. Hence $\mathcal{S}, \mathcal{E}^a, \mathcal{I}^a \not\models_{\rightsquigarrow} \varphi \to [a]\psi$, and then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{I}^a \rangle \not\models \varphi \to [a]\psi$. ∎

Proof of Theorem 8.6

If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ *satisfies Postulate* **PS***, *then* $\mathcal{D} \models \varphi \to \langle a \rangle \top$ *if and only if* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{X}^a \rangle \models \varphi \to \langle a \rangle \top$.

($\Rightarrow$): Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I} \rangle$ satisfy Postulate **PS***, and suppose that $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{X}^a \rangle \not\models \varphi \to \langle a \rangle \top$, i.e., $\mathcal{S}, \mathcal{X}^a \not\models_{\mathsf{PDL}} \varphi \to \langle a \rangle \top$. Then there exists a PDL-model $\mathscr{M} = \langle W, R_a \rangle$, such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{X}^a$ and $\not\models^{\mathscr{M}} \varphi \to \langle a \rangle \top$. This means that there is a possible world $v \in W$ such that $\models^{\mathscr{M}}_{v} \varphi$ and $\not\models^{\mathscr{M}}_{v} \langle a \rangle \top$.

(We extend $\mathscr{M}$ to build a model of $\mathcal{D}$ and then conclude that $\mathcal{D} \not\models \varphi \to \langle a \rangle \top$.)

Given $w \in W$, for each $a_i \in \mathfrak{Act}$ we define:

$$\mathcal{I}^{a_i}(w) = \{\varphi_j \to [a_i]\bot \in \mathcal{I}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

$$\mathcal{X}^{a_i}(w) = \{\varphi_j \to \langle a_i\rangle\top \in \mathcal{X}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

Let $\mathscr{M}' = \langle W', R'\rangle$ be such that $W' = W$, and $R' = R_a \cup \bigcup_{a' \neq a} R_{a'}$, where for each $a' \neq a$ and every $w, w' \in W'$, $wR_{a'}w'$ if and only if

- $\models^{\mathscr{M}'}_{w'} \neg\ell$ for all ℓ such that $a' \not\rightsquigarrow \ell$ and $\models^{\mathscr{M}'}_{w} \neg\ell$;
- $\models^{\mathscr{M}'}_{w'} \psi_i$ for every $\varphi_i \to [a']\psi_i \in \mathcal{E}^{a'}$ such that $\models^{\mathscr{M}'}_{w} \varphi_i$; and
- $\mathcal{I}^{a'}(w) = \emptyset$.

By definition, $\mathscr{M}'$ is a model of the dependence relation $\rightsquigarrow$. Because, by hypothesis, $\mathcal{D}$ satisfies **PS***, there is no implicit static law, i.e., for every $a_i \in \mathfrak{Act}$ and every $w \in W'$, if $\mathcal{X}^{a_i}(w) \neq \emptyset$, then $\mathcal{I}^{a_i}(w) = \emptyset$. Then, as $W' = \mathit{valuations}(\mathcal{S})$, $\mathscr{M}'$ is a model of $\mathcal{S}$. We have that $\mathscr{M}'$ is a model of $\mathcal{E}$, too: it is a model of $\mathcal{E}^a$, and given $a' \neq a$, for every $\varphi_i \to [a']\psi_i \in \mathcal{E}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\models^{\mathscr{M}'}_{w'} \psi_i$ for all $w' \in W'$ such that $wR_{a'}w'$. Clearly $\mathscr{M}'$ is also a model of $\mathcal{I}$: it is a model of $\mathcal{I}^a$, and given $a' \neq a$, for every $\varphi_i \to [a']\bot \in \mathcal{I}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\mathcal{I}^{a'}(w) \neq \emptyset$ and $R_{a'}(w) = \emptyset$. $\mathscr{M}'$ is a model of $\mathcal{X}$, too: besides being a model of $\mathcal{X}^a$, for every $a' \neq a$ and all worlds $w \in W'$ such that $\mathcal{X}^{a'}(w) \neq \emptyset$ there is a world accessible by $R_{a'}$, because $R_{a'}(w) = \emptyset$ in this case would preclude $\mathcal{X}^{a'}(w) \neq \emptyset$, and otherwise $w \notin W'$, which is impossible as long as **PS*** is satisfied. Hence $\models^{\mathscr{M}'} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$. Because there is $v \in W'$ such that $\models^{\mathscr{M}'}_{v} \varphi$ and $\not\models^{\mathscr{M}'}_{v} \langle a\rangle\top$, we have $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \to \langle a\rangle\top$, and then $\mathcal{D} \not\models \varphi \to \langle a\rangle\top$.

($\Leftarrow$): Suppose $\mathcal{D} \not\models \varphi \to \langle a\rangle\top$, i.e., $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \to \langle a\rangle\top$. Then there is a $\rightsquigarrow$-model $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$ and $\not\models^{\mathscr{M}} \varphi \to \langle a\rangle\top$. Then, given a, we have $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{X}^a \wedge \mathcal{I}^a$, and then $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{X}^a$. Moreover, by definition, $\mathscr{M}$ is a PDL-model. Hence $\mathcal{S}, \mathcal{X}^a \not\models_{\mathsf{PDL}} \varphi \to \langle a\rangle\top$, and then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\mathsf{PDL}}, \mathcal{S} \cup \mathcal{X}^a\rangle \not\models \varphi \to \langle a\rangle\top$. ∎

Proof of Theorem 8.8

If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I}\rangle$ *satisfies Postulate* **PS***, *then* $\mathcal{D} \models \varphi \to [a_1; \ldots; a_n]\psi$ *if and only if* $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n}\rangle \models \varphi \to [a_1; \ldots; a_n]\psi$.

Lemma D.1

If $\mathcal{D} \models \varphi \to [a_1; \ldots; a_n]\psi$, *then there is* $\varphi' \in \mathfrak{Fml}$ *such that* $\mathcal{D} \models \varphi \to [a_1; \ldots; a_{n-1}]\varphi'$ *and* $\mathcal{D} \models \varphi' \to [a_n]\psi$.

Proof:
Let $\mathcal{D} \models \varphi \rightarrow [a_1;\ldots;a_n]\psi$. If $\mathcal{D} \models \varphi \rightarrow [a_1;\ldots;a_n]\bot$, the result immediately follows. Then, given a model $\mathscr{M} = \langle W, R\rangle$ of $\mathcal{D}$ such that $\models^{\mathscr{M}}_{w} \varphi$ for some $w \in W$, if $\models^{\mathscr{M}}_{w} \langle a_1;\ldots;a_n\rangle\top$, there must be at least one w'_{n-1} such that $\models^{\mathscr{M}}_{w'_{n-1}} [a_n]\psi$. Take all such w'_{n-1} and let φ' be

$$\bigvee_{\models^{\mathscr{M}}_{w'_{n-1}} [a_n]\psi} w'_{n-1}$$

Then we have $\mathcal{D} \models \varphi \rightarrow [a_1;\ldots;a_{n-1}]\varphi'$, and $\mathcal{D} \models \varphi' \rightarrow [a_n]\psi$. ■

Proof of Theorem 8.8

($\Rightarrow$): The proof is by induction on the number of action operators.

Base: $n = 1$. As $\mathcal{D}$ satisfies Postulate **PS***, the result follows from Theorem 8.5.

Induction hypothesis: for any $k < n$, if $\mathcal{D} \models \varphi \rightarrow [a_1;\ldots;a_k]\psi$, then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_k} \cup \mathcal{I}^{a_1,\ldots,a_k}\rangle \models \varphi \rightarrow [a_1;\ldots;a_k]\psi$.

Step: let $\mathcal{D} \models \varphi \rightarrow [a_1;\ldots;a_n]\psi$. By Lemma D.1, there is a classical formula φ' such that $\mathcal{D} \models \varphi \rightarrow [a_1;\ldots;a_{n-1}]\varphi'$ and $\mathcal{D} \models \varphi' \rightarrow [a_n]\psi$. From the induction hypothesis, we have that $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_{n-1}} \cup \mathcal{I}^{a_1,\ldots,a_{n-1}}\rangle \models \varphi \rightarrow [a_1;\ldots;a_{n-1}]\varphi'$ and $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_n} \cup \mathcal{I}^{a_n}\rangle \models \varphi' \rightarrow [a_n]\psi$. This gives us $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n}\rangle \models \varphi \rightarrow [a_1;\ldots;a_n]\psi$.

($\Leftarrow$): Suppose $\mathcal{D} \not\models \varphi \rightarrow [a_1;\ldots;a_n]\psi$, i.e., $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \rightarrow [a_1;\ldots;a_n]\psi$. Then there is a $\rightsquigarrow$-model $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$ and $\not\models^{\mathscr{M}} \varphi \rightarrow [a_1;\ldots;a_n]\psi$. Then, given $a_1,\ldots,a_n$, we have $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^{a_1,\ldots,a_n} \wedge \mathcal{X}^{a_1,\ldots,a_n} \wedge \mathcal{I}^{a_1,\ldots,a_n}$, and then $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^{a_1,\ldots,a_n} \wedge \mathcal{I}^{a_1,\ldots,a_n}$. Hence $\mathcal{S}, \mathcal{E}^{a_1,\ldots,a_n}, \mathcal{I}^{a_1,\ldots,a_n} \not\models_{\rightsquigarrow} \varphi \rightarrow [a_1;\ldots;a_n]\psi$, and then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n}\rangle \not\models \varphi \rightarrow [a_1;\ldots;a_n]\psi$. ■

Proof of Theorem 8.9

If $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I}\rangle$ satisfies Postulate **PS***, then $\mathcal{D} \models \varphi \rightarrow \langle a_1;\ldots;a_n\rangle\psi$ if and only if $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{X}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n}\rangle \models \varphi \rightarrow \langle a_1;\ldots;a_n\rangle\psi$.*

Lemma D.2
Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \cup \mathcal{I}\rangle$ satisfy Postulate **PS***. If $\mathcal{D} \models \varphi \rightarrow \langle a\rangle\psi$ is the case, then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a} \cup \mathcal{X}^{a} \cup \mathcal{I}^{a}\rangle \models \varphi \rightarrow \langle a\rangle\psi$.*

Proof:
Let $\mathcal{D}$ satisfy Postulate **PS*** and suppose $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^a \cup \mathcal{X}^a \cup \mathcal{I}^a \rangle \not\models \varphi \rightarrow \langle a \rangle \psi$, i.e., $\mathcal{S}, \mathcal{E}^a, \mathcal{X}^a, \mathcal{I}^a \not\models_{\rightsquigarrow} \varphi \rightarrow \langle a \rangle \psi$. Then there exists a $\rightsquigarrow$-model $\mathscr{M} = \langle W, R_a \rangle$, such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^a \wedge \mathcal{X}^a \wedge \mathcal{I}^a$ and $\not\models^{\mathscr{M}} \varphi \rightarrow \langle a \rangle \psi$. This means that there is a possible world $v \in W$ such that $\models^{\mathscr{M}}_{v} \varphi$ and $\not\models^{\mathscr{M}}_{v} \langle a \rangle \psi$.

(We extend $\mathscr{M}$ to build a model of $\mathcal{D}$ and then conclude that $\mathcal{D} \not\models \varphi \rightarrow \langle a \rangle \psi$.)

Given $w \in W$, for each $a_i \in \mathfrak{Act}$ we define:

$$\mathcal{I}^{a_i}(w) = \{\varphi_j \rightarrow [a_i]\bot \in \mathcal{I}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

$$\mathcal{X}^{a_i}(w) = \{\varphi_j \rightarrow \langle a_i \rangle \top \in \mathcal{X}^{a_i} \ : \models^{\mathscr{M}}_{w} \varphi_j\}$$

Let $\mathscr{M}' = \langle W', R' \rangle$ be such that $W' = W$, and $R' = R_a \cup \bigcup_{a' \neq a} R_{a'}$ (we extend $\mathscr{M}$ to all other actions $\mathcal{D}$ speaks of), where for each $a' \neq a$ and every $w, w' \in W'$, $wR_{a'}w'$ if and only if

- $\models^{\mathscr{M}'}_{w'} \neg \ell$ for all ℓ such that $a' \not\rightsquigarrow \ell$ and $\models^{\mathscr{M}'}_{w} \neg \ell$;
- $\models^{\mathscr{M}'}_{w'} \psi_i$ for every $\varphi_i \rightarrow [a']\psi_i \in \mathcal{E}^{a'}$ such that $\models^{\mathscr{M}'}_{w} \varphi_i$; and
- $\mathcal{I}^{a'}(w) = \emptyset$.

By definition, $\mathscr{M}'$ is a model of the dependence relation $\rightsquigarrow$. Because, by hypothesis, $\mathcal{D}$ satisfies **PS***, there is no implicit static law, i.e., for every $a_i \in \mathfrak{Act}$ and every $w \in W'$, if $\mathcal{X}^{a_i}(w) \neq \emptyset$, then $\mathcal{I}^{a_i}(w) = \emptyset$. Then, as $W' = \mathit{valuations}(\mathcal{S})$, $\mathscr{M}'$ is a model of $\mathcal{S}$. We have that $\mathscr{M}'$ is a model of $\mathcal{E}$, too: it is a model of $\mathcal{E}^a$, and given $a' \neq a$, for every $\varphi_i \rightarrow [a']\psi_i \in \mathcal{E}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\models^{\mathscr{M}'}_{w'} \psi_i$ for all $w' \in W'$ such that $wR_{a'}w'$. Clearly $\mathscr{M}'$ is also a model of $\mathcal{I}$: it is a model of $\mathcal{I}^a$, and given $a' \neq a$, for every $\varphi_i \rightarrow [a']\bot \in \mathcal{I}$ and every $w \in W'$, if $\models^{\mathscr{M}'}_{w} \varphi_i$, then $\mathcal{I}^{a'}(w) \neq \emptyset$ and $R_{a'}(w) = \emptyset$. $\mathscr{M}'$ is a model of $\mathcal{X}$, too: besides being a model of $\mathcal{X}^a$, for every $a' \neq a$ and all worlds $w \in W'$ such that $\mathcal{X}^{a'}(w) \neq \emptyset$ there is a world accessible by $R_{a'}$, because $R_{a'}(w) = \emptyset$ in this case would preclude $\mathcal{X}^{a'}(w) \neq \emptyset$, and otherwise $w \notin W'$, which is impossible as long as **PS*** is satisfied. Hence $\models^{\mathscr{M}'} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$. Because there is $v \in W'$ such that $\models^{\mathscr{M}'}_{v} \varphi$ and $\not\models^{\mathscr{M}'}_{v} \langle a \rangle \psi$, we have $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \rightarrow \langle a \rangle \psi$, and then $\mathcal{D} \not\models \varphi \rightarrow \langle a \rangle \top$. ∎

Lemma D.3
If $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_n \rangle \psi$, then there is $\varphi' \in \mathfrak{Fml}$ such that $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_{n-1} \rangle \varphi'$ and $\mathcal{D} \models \varphi' \rightarrow \langle a_n \rangle \psi$.

Proof:

The proof is by induction on the number of action operators.

Base: $n = 2$. Suppose $\mathcal{D} \models \varphi \rightarrow \langle a_1; a_2\rangle\psi$. Then $\mathcal{D} \models \varphi \rightarrow \langle a_1\rangle\langle a_2\rangle\psi$. For every model $\mathscr{M} = \langle W, R\rangle$ of $\mathcal{D}$ and for every $w \in W$ such that $\models^{\mathscr{M}}_{w} \varphi$, there is $w' \in W$ such that $wR_{a_1}w'$ and $\models^{\mathscr{M}}_{w'}\langle a_2\rangle\psi$. Let φ' be $\bigwedge\{\ell : \ell \in w'\}$ and the result follows.

Induction hypothesis: for any $k < n$, if $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_k\rangle\psi$, then there is $\varphi' \in \mathfrak{Fml}$ such that $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_{k-1}\rangle\varphi'$ and $\mathcal{D} \models \varphi' \rightarrow \langle a_k\rangle\psi$.

Step: let $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$. Then $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_{n-1}\rangle\top$. By the induction hypothesis, there is $\varphi' \in \mathfrak{Fml}$ such that $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_{n-2}\rangle\varphi'$ and $\mathcal{D} \models \varphi' \rightarrow \langle a_{n-1}\rangle\top$. Because $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$, given a model $\mathscr{M} = \langle W, R\rangle$ of $\mathcal{D}$ such that $\models^{\mathscr{M}}_{w} \varphi$ for some $w \in W$, there must be $w'_{n-2} \in W$ such that $\models^{\mathscr{M}}_{w'_{n-2}} \langle a_{n-1}\rangle\langle a_n\rangle\psi$. Then we can safely take φ' as $\bigwedge\{\ell : \ell \in w'_{n-2}\}$. Now, $\mathcal{D} \models \varphi' \rightarrow \langle a_{n-1}\rangle\langle a_n\rangle\psi$. By the base step, there is $\varphi'' \in \mathfrak{Fml}$ such that $\mathcal{D} \models \varphi' \rightarrow \langle a_{n-1}\rangle\varphi''$ and $\mathcal{D} \models \varphi'' \rightarrow \langle a_n\rangle\psi$. Putting all the results together, we get $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_{n-1}\rangle\varphi''$ and $\mathcal{D} \models \varphi'' \rightarrow \langle a_n\rangle\psi$, for some $\varphi'' \in \mathfrak{Fml}$. ■

Proof of Theorem 8.9

($\Rightarrow$): The proof is by induction on the number of action operators.

Base: $n = 1$. As $\mathcal{D}$ satisfies Postulate **PS***, the result follows from Lemma D.2.

Induction hypothesis: for any $k < n$, if $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_k\rangle\psi$, then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_k} \cup \mathcal{X}^{a_1,\ldots,a_k} \cup \mathcal{I}^{a_1,\ldots,a_k}\rangle \models \varphi \rightarrow \langle a_1; \ldots; a_k\rangle\psi$.

Step: let $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$. By Lemma D.3, there is $\varphi' \in \mathfrak{Fml}$ such that $\mathcal{D} \models \varphi \rightarrow \langle a_1; \ldots; a_{n-1}\rangle\varphi'$ and $\mathcal{D} \models \varphi' \rightarrow \langle a_n\rangle\psi$. By the induction hypothesis, we have $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_{n-1}} \cup \mathcal{X}^{a_1,\ldots,a_{n-1}} \cup \mathcal{I}^{a_1,\ldots,a_{n-1}}\rangle \models \varphi \rightarrow \langle a_1; \ldots; a_{n-1}\rangle\varphi'$ and also $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_n} \cup \mathcal{X}^{a_n} \cup \mathcal{I}^{a_n}\rangle \models \varphi' \rightarrow \langle a_n\rangle\psi$. Then, this gives us $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{X}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n}\rangle \models \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$.

($\Leftarrow$): Suppose $\mathcal{D} \not\models \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$, i.e., $\mathcal{S}, \mathcal{E}, \mathcal{X}, \mathcal{I} \not\models_{\rightsquigarrow} \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$. Then there is a $\rightsquigarrow$-model $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X} \wedge \mathcal{I}$ and $\not\models^{\mathscr{M}} \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$. Then, given $a_1, \ldots, a_n$, we have $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^{a_1,\ldots,a_n} \wedge \mathcal{X}^{a_1,\ldots,a_n} \wedge \mathcal{I}^{a_1,\ldots,a_n}$, and hence $\mathcal{S}, \mathcal{E}^{a_1,\ldots,a_n}, \mathcal{X}^{a_1,\ldots,a_n}, \mathcal{I}^{a_1,\ldots,a_n} \not\models_{\rightsquigarrow} \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$. Then $\langle \mathcal{L}_{\mathsf{PDL}}, \models_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E}^{a_1,\ldots,a_n} \cup \mathcal{X}^{a_1,\ldots,a_n} \cup \mathcal{I}^{a_1,\ldots,a_n}\rangle \not\models \varphi \rightarrow \langle a_1; \ldots; a_n\rangle\psi$. ■

Long Proofs of Chapter 9

Proof of Theorem 9.1

Let Φ be a formula that has the form of one of the three laws. For all models $\mathscr{M}'$, if $\mathscr{M}' \in \mathscr{M}_{\Phi}^{-}$ for some $\mathscr{M} = \langle W, R\rangle$ such that $\models^{\mathscr{M}} \mathcal{D}$, then $\models^{\mathscr{M}'} \mathcal{D}_{\Phi}^{-}$.

Let $\mathscr{M}$ be such that $\models^{\mathscr{M}} \mathcal{D}$ and let $\mathscr{M}' \in \mathscr{M}_{\Phi}^{-}$. We analyze each case.

Suppose Φ is φ, for some propositional $\varphi \in \mathfrak{Fml}$. Then $\mathscr{M}' = \langle W', R\rangle$, where $W' = W \ominus \mathit{valuations}(\varphi)$. Because we have assumed the syntactical classical contraction operator $\ominus$ is sound and complete w.r.t. its semantics, $\mathscr{M}'$ is a model of $\mathcal{S}^{-}$. As $\rightsquigarrow$ and $\mathcal{E}$ have not changed, clearly $\mathscr{M}'$ remains a $\rightsquigarrow$-model and a model of $\mathcal{E}$. $\mathscr{M}'$ is also a model of $\mathcal{X}^{-}$: for every $w \in W'$ and every $(\varphi_i \wedge \varphi) \to \langle a\rangle\top \in \mathcal{X}^{-}$, $\models_{w}^{\mathscr{M}'} \varphi_i \wedge \varphi$ implies $R_a(w) \neq \emptyset$, because $\models_{w}^{\mathscr{M}} \varphi_i \to \langle a\rangle\top$. Hence $\models^{\mathscr{M}'} \mathcal{S}^{-} \wedge \mathcal{E} \wedge \mathcal{X}^{-}$, and then $\models^{\mathscr{M}'} \mathcal{D}_{\varphi}^{-}$.

Let now Φ have the form $\varphi \to [a]\psi$, for $\varphi, \psi \in \mathfrak{Fml}$. Then $\mathscr{M}' = \langle W, R \cup R_a'\rangle$ such that $R_a' \subseteq \{(w, w') \ : \models_{w}^{\mathscr{M}} \varphi\}$. It is enough to show that $\mathscr{M}'$ is a model of $\mathcal{E}^{-}$ and of the new dependence relation $\rightsquigarrow'$. Clearly it is a model of $\rightsquigarrow'$, since it is a $\rightsquigarrow$-model and $\rightsquigarrow \subseteq \rightsquigarrow'$. Now, for all $w \in W$ and every $(\varphi_i \wedge \neg\varphi) \to [a]\psi_i \in \mathcal{E}^{-}$, if $\models_{w}^{\mathscr{M}'} \varphi_i \wedge \neg\varphi$, then $\models_{w}^{\mathscr{M}'} \varphi_i$, from what it follows $\models_{w}^{\mathscr{M}} \varphi_i$, and because $\models^{\mathscr{M}} \mathcal{E}$, $\models_{w'}^{\mathscr{M}} \psi_i$ for all $w' \in R_a(w)$. Moreover, as $\not\models_{w}^{\mathscr{M}'} \varphi$, we have $\not\models_{w}^{\mathscr{M}} \varphi$, and then $R_a'(w) = \emptyset$. Putting both results together, it follows $\models_{w'}^{\mathscr{M}'} \psi_i$ for all $w' \in R_a(w)$, and then $\models^{\mathscr{M}'} \mathcal{E}^{-}$. Hence $\models^{\mathscr{M}'} \mathcal{D}_{\varphi \to [a]\psi}^{-}$.

Now let Φ be of the form $\varphi \to \langle a\rangle\top$, for some $\varphi \in \mathfrak{Fml}$. Then $\mathscr{M}' = \langle W, R \setminus R_a'\rangle$, such that $R_a' \subseteq \{(w, w') \ : \ wR_aw' \text{ and } \models_{w}^{\mathscr{M}} \varphi\}$. It suffices to show that $\mathscr{M}'$ is a model of $\mathcal{X}^{-}$. For all $w \in W$ and every $(\varphi_i \wedge \neg\varphi) \to \langle a\rangle\top \in \mathcal{X}^{-}$, if $\models_{w}^{\mathscr{M}'} \varphi_i \wedge \neg\varphi$, then $\models_{w}^{\mathscr{M}'} \varphi_i$, from what it follows $\models_{w}^{\mathscr{M}} \varphi_i$, and because $\models^{\mathscr{M}} \mathcal{X}$, there exists $w' \in W$ such that wR_aw'. Because $\not\models_{w}^{\mathscr{M}'} \varphi$, $\not\models_{w}^{\mathscr{M}} \varphi$, and then $R_a'(w) = \emptyset$. Putting both results together, it follows $\models_{w}^{\mathscr{M}'} \langle a\rangle\top$, and thus $\models^{\mathscr{M}'} \mathcal{X}^{-}$. Hence $\models^{\mathscr{M}'} \mathcal{D}_{\varphi \to \langle a\rangle\top}^{-}$. ∎

Proof of Lemma 9.2

Let $\mathcal{D} = \langle \mathcal{L}_{\mathsf{PDL}}, \mid\!\!\sim_{\rightsquigarrow}, \mathcal{S} \cup \mathcal{E} \cup \mathcal{X} \rangle$ be modular, and let Φ be a formula of the form of one of the three laws. Then $\mathcal{D}^{-}_{\Phi}$ is modular.

We analyze each case.

Let Φ be φ, for some propositional $\varphi \in \mathfrak{Fml}$, and suppose $\mathcal{D}^{-}_{\varphi}$ is not modular. Then there exists $\varphi' \in \mathfrak{Fml}$ such that $\mathcal{D}^{-}_{\varphi} \models \varphi'$ and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S}^{-} \rangle \not\models \varphi'$, i.e., $\mathcal{S}^{-}, \mathcal{E}, \mathcal{X}^{-} \mid\!\!\sim_{\rightsquigarrow} \varphi'$ and $\mathcal{S}^{-} \not\models_{\mathsf{CPL}} \varphi'$. As the original $\mathcal{X}$ has been weakened and the syntactical propositional contraction operator $\ominus$ has been assumed to satisfy Katsuno and Mendelzon's postulate $Cn(\mathcal{S} \ominus \varphi) \subseteq Cn(\mathcal{S})$, we must have $\mathcal{S}, \mathcal{E}, \mathcal{X} \mid\!\!\sim_{\rightsquigarrow} \varphi'$. Because $\mathcal{D}$ is modular, it holds $\mathcal{S} \models_{\mathsf{CPL}} \varphi'$. Then we have at least $\mathit{valuations}(\neg\varphi') \subseteq \mathit{valuations}(\neg\varphi)$, for $\mathcal{S}^{-} \not\models_{\mathsf{CPL}} \varphi'$. This means $\mathcal{S}, \mathcal{E}, \mathcal{X} \mid\!\!\sim_{\rightsquigarrow} \varphi \rightarrow \varphi'$, and then $\ominus$ has not worked as expected.

Let now Φ have the form $\varphi \rightarrow [a]\psi$, for $\varphi, \psi \in \mathfrak{Fml}$, and suppose $\mathcal{D}^{-}_{\varphi \rightarrow [a]\psi}$ is not modular. Then there exists $\varphi' \in \mathfrak{Fml}$ such that $\mathcal{D}^{-}_{\varphi \rightarrow [a]\psi} \models \varphi'$ and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \rangle \not\models \varphi'$, i.e., $\mathcal{S}, \mathcal{E}^{-}, \mathcal{X} \mid\!\!\sim_{\rightsquigarrow'} \varphi'$ and $\mathcal{S} \not\models_{\mathsf{CPL}} \varphi'$.

Claim: If $\mathcal{S}, \mathcal{E}^{-}, \mathcal{X} \mid\!\!\sim_{\rightsquigarrow'} \varphi'$, then $\mathcal{S}, \mathcal{E}^{-}, \mathcal{X} \mid\!\!\sim_{\rightsquigarrow} \varphi'$.

(Proof of the claim): Straightforward: suppose $\mathcal{S}, \mathcal{E}^{-}, \mathcal{X} \not\mid\!\!\sim_{\rightsquigarrow} \varphi'$. Then there exists a possible worlds model $\mathscr{M} = \langle W, R \rangle$ such that $\mathscr{M}$ is a $\rightsquigarrow$-model, $\dot{\models}^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^{-} \wedge \mathcal{X}$, and $\dot{\not\models}^{\mathscr{M}} \varphi'$. Because $\rightsquigarrow \subseteq \rightsquigarrow'$, $\mathscr{M}$ is a $\rightsquigarrow'$-model, too. Hence, $\mathcal{S}, \mathcal{E}^{-}, \mathcal{X} \not\mid\!\!\sim_{\rightsquigarrow'} \varphi'$. (End of the proof of the claim.)

Claim: $\mathcal{S}, \mathcal{E}, \mathcal{X} \mid\!\!\sim_{\rightsquigarrow} \mathcal{S} \wedge \mathcal{E}^{-} \wedge \mathcal{X}$.

(Proof of the claim): We show that there is no $\rightsquigarrow$-model $\mathscr{M}$ such that $\dot{\models}^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}$ and $\dot{\not\models}^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^{-} \wedge \mathcal{X}$. Let $\mathscr{M} = \langle W, R \rangle$ be a $\rightsquigarrow$-model such that $\dot{\not\models}^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E}^{-} \wedge \mathcal{X}$. Then there exists $w \in W$ such that $\dot{\not\models}^{\mathscr{M}}_{w} \mathcal{S} \wedge \mathcal{E}^{-} \wedge \mathcal{X}$. If $\dot{\not\models}^{\mathscr{M}}_{w} \mathcal{S}$ or $\dot{\not\models}^{\mathscr{M}}_{w} \mathcal{X}$, the result follows. Consider $\dot{\not\models}^{\mathscr{M}}_{w} \mathcal{E}^{-}$. Then, there is some $\widehat{\mathcal{E}^{-}} \subseteq \mathcal{E}^{-}$ such that

$$\dot{\models}^{\mathscr{M}}_{w} \bigwedge_{(\varphi_i \wedge \neg\varphi) \rightarrow [a]\psi_i \in \widehat{\mathcal{E}^{-}}} (\varphi_i \wedge \neg\varphi)$$

and there exists $w' \in W$ such that $wR_a w'$ and

$$\dot{\not\models}^{\mathscr{M}}_{w'} \bigwedge_{(\varphi_i \wedge \neg\varphi) \rightarrow [a]\psi_i \in \widehat{\mathcal{E}^{-}}} \psi_i$$

Then

$$\models^{\mathscr{M}}_{w} \bigwedge_{(\varphi_i \wedge \neg\varphi)\to[a]\psi_i \in \mathcal{E}^{\widehat{-}}} \varphi_i$$

Taking the obvious corresponding $\hat{\mathcal{E}} \subseteq \mathcal{E}$, we get

$$\models^{\mathscr{M}}_{w} \bigwedge_{\varphi_i \to [a]\psi_i \in \hat{\mathcal{E}}} \varphi_i \text{ and } \not\models^{\mathscr{M}}_{w'} \bigwedge_{\varphi_i \to [a]\psi_i \in \hat{\mathcal{E}}} \psi_i$$

Hence, $\not\models^{\mathscr{M}}_{w} \mathcal{E}$, and then $\not\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}$. (End of the proof of the claim.)

From $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \mathcal{S} \wedge \mathcal{E}^{-} \wedge \mathcal{X}$, and $\mathcal{S}, \mathcal{E}^{-}, \mathcal{X} \models_{\rightsquigarrow} \varphi'$, it follows $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \varphi'$. Because $\mathcal{S} \not\models_{\mathsf{CPL}} \varphi'$, $\mathcal{D}$ is not modular.

Now suppose Φ has the form $\varphi \to \langle a \rangle \top$, for some $\varphi \in \mathfrak{Fml}$, and suppose $\mathcal{D}^{-}_{\varphi \to \langle a \rangle \top}$ is not modular. Then there exists $\varphi' \in \mathfrak{Fml}$ such that we have $\mathcal{D}^{-}_{\varphi \to \langle a \rangle \top} \models \varphi'$ and $\langle \mathcal{L}_{\mathsf{CPL}}, \models_{\mathsf{CPL}}, \mathcal{S} \rangle \not\models \varphi'$, i.e., $\mathcal{S}, \mathcal{E}, \mathcal{X}^{-} \models_{\rightsquigarrow} \varphi'$ and $\mathcal{S} \not\models_{\mathsf{CPL}} \varphi'$.

Claim: $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}^{-}$.

(Proof of the claim): We show that there is no $\rightsquigarrow$-model $\mathscr{M}$ such that $\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}$ and $\not\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}^{-}$. Let $\mathscr{M} = \langle W, R \rangle$ be a $\rightsquigarrow$-model such that $\not\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}^{-}$. Then there exists $w \in W$ such that $\not\models^{\mathscr{M}}_{w} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}^{-}$. If $\not\models^{\mathscr{M}}_{w} \mathcal{S}$ or $\not\models^{\mathscr{M}}_{w} \mathcal{E}$, the result follows. Consider $\not\models^{\mathscr{M}}_{w} \mathcal{X}^{-}$. Then, there is some $\hat{\mathcal{X}}^{-} \subseteq \mathcal{X}^{-}$ such that

$$\models^{\mathscr{M}}_{w} \bigwedge_{(\varphi_i \wedge \neg\varphi)\to\langle a\rangle\top \in \hat{\mathcal{X}}^{-}} (\varphi_i \wedge \neg\varphi)$$

and $R_a(w) = \emptyset$. Then

$$\models^{\mathscr{M}}_{w} \bigwedge_{(\varphi_i \wedge \neg\varphi)\to\langle a\rangle\top \in \mathcal{X}^{\widehat{-}}} \varphi_i$$

Taking the obvious corresponding $\hat{\mathcal{X}} \subseteq \mathcal{X}$, we get

$$\models^{\mathscr{M}}_{w} \bigwedge_{\varphi_i \to \langle a \rangle \top \in \hat{\mathcal{X}}} \varphi_i$$

Because $R_a(w) = \emptyset$, $\not\models^{\mathscr{M}}_{w} \mathcal{X}$, and then $\not\models^{\mathscr{M}} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}$. (End of the proof of the claim.)

From $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \mathcal{S} \wedge \mathcal{E} \wedge \mathcal{X}^{-}$, and $\mathcal{S}, \mathcal{E}, \mathcal{X}^{-} \models_{\rightsquigarrow} \varphi'$, it follows $\mathcal{S}, \mathcal{E}, \mathcal{X} \models_{\rightsquigarrow} \varphi'$. Because $\mathcal{S} \not\models_{\mathsf{CPL}} \varphi'$, $\mathcal{D}$ is not modular. ∎

Ceci n'est pas un résumé

I love French wine, like I love the French language. I have sampled every language, French is my favourite – fantastic – language, especially to curse with. Nom de Dieu de putain de bordel de merde de saloperie de connard d'enculé de ta mère ! You see, it's like wiping your ass with silk, I love it.

— Merovingian, in Matrix Reloaded

Pourquoi on est là ?

Dans les approches de représentation de connaissances basées sur la logique, la connaissance concernant un domaine est habituellement décrite par des ensembles de formules logiques (théories). Dans le raisonnement sur les actions nous nous intéressons à des théories qui décrivent le comportement d'actions sur des propriétés du monde. Nous appelons de telles théories *théories d'action* ou *descriptions de domaine*.

D'habitude, une théorie d'actions est un ensemble d'énoncés ayant la forme : " si *contexte*, alors *effet* après *chaque exécution* d'*action* " ; et " si *pré-condition*, alors *action exécutable* ". Le premier type d'énoncé est utilisé pour exprimer des lois d'effet, c'est-à-dire des formules qui relient une action à son résultat, étant donné un contexte particulier. Le second type d'énoncé représente des lois d'exécutabilité, qui établissent les conditions suffisantes sous lesquelles une action est exécutable. Leur dual fournit les conditions nécessaires pour qu'une action soit exécutable : " si *pré-condition*, alors *action impossible* ". Finalement, dans une représentation d'un domaine dynamique, nous avons également des énoncés qui ne mentionnent aucune action. Ceux-ci représentent des lois sur la partie *statique* du monde, c'est-à-dire les contraintes qui déterminent les états possibles. Nous appelons ce type de contraintes les lois statiques du domaine.

Lorsque l'on décrit des théories d'action, l'objectif est de doter un agent de la capacité à raisonner dans un domaine dynamique et d'agir de manière rationnelle. Parmi les différents types de raisonnement qu'un agent peut avoir, nous identifions la vérification de la *consistance* de sa théorie ; la *prévision* des effets des actions ;

l'*explication* de l'observation d'un effet donné ; l'élaboration d'un *plan* pour accomplir un but ; la vérification de l'*exécutabilité* ou de l'*inexécutabilité* d'une action donnée ; et la *révision* et *mise à jour* de ses connaissances sur le comportement d'une action.

A priori la consistance est le seul critère fourni par la logique formelle pour vérifier la qualité des théories d'action. Dans ce travail, nous arguons que toutes les approches existantes dans la littérature sont très libérales dans le sens où nous pouvons avoir des théories d'action satisfiables qui sont intuitivement incorrectes. Donc une notion au-delà de la consistance est nécessaire pour évaluer des descriptions de domaine.

Le modulaire c'est le bon

La modularité est devenue un des mots d'ordre de nombreux domaines en informatique. C'est aussi le cas pour la représentation de la connaissance et le raisonnement, où des descriptions monolithiques se sont montrées d'utilisation très complexes.

Les dernières années ont vue la parution de plusieurs travaux qui, de manière plus ou moins implicite, abordent des concepts tels que module, intelligibilité, évaluation, tolérance à l'élaboration, et d'autres. La plupart de ces termes sont empruntés de l'ingénierie de logiciel, parfois sans une notion claire des impacts qu'ils peuvent avoir quand ils sont transposés dans des domaines où leur emploie n'est pas une question d'intuition, mais ils doivent plutôt s'accommoder avec des cadres formels bien établis. Dans ce travail nous montrons que ce n'est pas une tâche simple, en particulier lorsque la logique constitue le cadre formel dans lequel la connaissance est représentée.

Ainsi, la question qui émerge naturellement est " comment pouvons-nous faciliter la tâche de l'ingénieur de connaissances pour décrire un domaine " ? Une réponse, en suivant évidemment la tendance diviser-et-conquérir, est la " modularisation de la théorie d'action ". Par contre, de même qu'en programmation orientée objet, modulariser une théorie d'action ne s'agit pas seulement de la découper en plusieurs morceaux. Un tel découpage doit être fait de sorte à ce que la théorie résultante ait des propriétés intéressantes. Dans cette thèse nous montrons que pour être considérés comme de bons modules, ils doivent satisfaire certains desiderata.

Quoi, dinde immortelle ?!

Considérons un scénario où nous avons une dinde et quelqu'un qui peut la mettre en marche ou bien lui tirer dessus avec un revolver. On peut formaliser ce scénario

à travers des formules logiques qui disent que " si la dinde est en marche, alors elle est vivante ", " si l'arme est chargée, alors après tirer la dinde meurt ", " en attirant la dinde elle se mets à marcher ", " si la dinde est morte, alors l'attirer ne la ressuscite pas ", et " on peut toujours attirer la dinde ".

Maintenant, du fait que " attirer la dinde la fait marcher " et " une dinde qui marche est vivante " on conclut " après attirer la dinde, elle est vivante ". A partir de ça et de l'information " si la dinde est morte, alors l'attirer ne la ressuscite pas ", on déduit que " si la dinde est morte, alors après l'attirer elle est à la fois vivante et morte ", c'est-à-dire une contradiction. Donc il n'est pas possible d'attirer une dinde morte. De ça et étant donné que c'est toujours possible d'attirer la dinde, on déduit que la dinde n'est jamais morte, c'est-à-dire, elle est immortelle !

Cet exemple, aussi simple soit il, illustre bien un problème important qui peut arriver dans des descriptions de domaine en raisonnement sur les actions : des interactions imprévues entre des formules. La présence de telles conséquences sont un signe de que la théorie d'action n'a pas été bien spécifiée. Dans notre exemple, le problème est dû au fait qu'on avait dit que c'était toujours possible d'attirer la dinde, ou alors au fait qu'on n'a pas complètement spécifié le contexte où l'action d'attirer la dinde la mets vraiment en marche.

Dans ce travail nous énonçons des postulats que toute théorie d'action doit satisfaire pour éviter que des situations comme celle ci-dessus se produisent. En plus, nous concevons aussi des algorithmes qui aident le concepteur du système à décider si une description de domaine satisfait l'ensemble de postulats et lui permettent de découvrir quelle partie de la théorie doit être modifiée pour la corriger.

Au delà du côté intuition, nous montrons aussi que des théories modulaires dans notre sens possèdent des propriétés computationelles intéressantes.

Il faut bien pouvoir changer la théorie

Supposons une situation où un agent a toujours cru que si l'interrupteur est en haut, alors il y a de la lumière dans la chambre. Supposons maintenant qu'un jour il observe que même si l'interrupteur est dans la position supérieure, la lumière est éteinte. Dans un tel cas, l'agent doit changer sa théorie au sujet de la relation entre les propositions " l'interrupteur est en haut " et " il y a de la lumière ". Cet exemple est une instance du problème de changement des bases de croyance propositionnelles, et il est largement abordé dans la littérature sur la révision et la mise à jour de croyances.

Ensuite, supposons que notre agent croit que chaque fois que l'interrupteur est en bas, après l'avoir basculé, il y a de la lumière dans la chambre. Ceci signifie que si la lumière est éteinte, dans chaque état du monde qui suit l'exécution de basculement de l'interrupteur, la chambre est éclairée. Puis, pendant une panne, l'agent bascule l'interrupteur et la chambre reste étonnamment dans l'obscurité.

Pour compléter les expériences de notre agent dans la découverte du comportement du monde, supposons qu'il a cru qu'il est toujours possible de basculer l'interrupteur, étant donnée la satisfaction de certaines conditions comme être assez proche de lui, avoir une main libre, l'interrupteur n'est pas cassé, etc. Cependant, un beau jour l'agent découvre que quelqu'un a mis de la colle sur l'interrupteur et, par conséquent, il n'est plus possible de le basculer.

Les derniers exemples illustrent des situations où le changement de croyances sur le comportement de l'action de basculer l'interrupteur est obligatoire. Dans le premier, basculer l'interrupteur, d'abord vu comme étant déterministe, doit maintenant être vu comme étant non déterministe, ou de manière alternative vu comme ayant des résultats différents dans un contexte spécifique (par exemple, si la centrale électrique est surchargée). Dans le deuxième exemple, l'exécutabilité de l'action considérée est remise en question à la lumière d'une nouvelle information montrant un contexte inconnu qui empêche son exécution.

De tels cas de changement de théorie sont très importants quand on manipule des descriptions logiques de domaines dynamiques : il peut toujours arriver qu'on découvre qu'une action a en fait un comportement différent de celui qu'on a toujours cru qu'elle avait.

Jusqu'ici, le changement de théorie a été étudié principalement pour les bases de connaissances dans les logiques classiques, en termes de révision et de mise à jour. Dans ce travail nous faisons donc un pas vers le changement de lois d'actions et proposons un cadre qui traite la mise à jour des théories d'action.

Or, qu'avons-nous fait ?

Notre contribution est double : générale, car nous présentons des postulats qui s'appliquent à tout formalisme en raisonnement sur les actions ; et spécifique, car nous proposons des algorithmes pour une solution existante au problème du décor.

Dans cette thèse nous avons identifié et fait une critique des approches principales de la modularité pour des descriptions de domaine, en précisant leurs caractéristiques

et en montrant pourquoi elles ne capturent pas complètement la modularité dans le sens nécessaire aux descriptions en raisonnement sur les actions. Nous avons argué que la modularité telle qu'utilisée usuellement en programmation ou définie dans les travaux sur la logique formelle n'est pas appropriée dans le raisonnement sur les actions. Dans le premier cas, ceci est en raison des restrictions d'expressivité. Dans le second cas, c'est parce que la modularité des théories logiques est habituellement trop forte et elle ne se montre pas très utile si la théorie est une description d'un scénario dans le raisonnement sur les actions.

Nous définissons donc notre concept de modularité pour les théories d'action et mettons en évidence les problèmes qui surgissent s'il n'est pas satisfait. En particulier, nous arguons que la partie non-dynamique des théories d'action pourrait influencer mais ne devrait pas être influencée par la partie dynamique.

Nous avons proposé quelques postulats, et en particulier nous avons essayé de démontrer que lorsqu'il y a des lois implicites, alors on s'est planté en concevant la théorie d'action en question. Comme montré, une solution possible découle de nos algorithmes, qui peuvent nous donner quelques directives lors de la correction d'une théorie d'action si nécessaire. Au moyen d'exemples, nous avons vu qu'il y a plusieurs alternatives de correction, et choisir le bon module à modifier aussi bien que fournir l'information intuitive qui doit être ajoutée est au concepteur du système.

Dans ce travail, nous avons illustré par quelques exemples ce que nous pouvons faire pour rendre une théorie intuitive. Ceci implique la modification de la théorie. Nous avons présenté une méthode générale pour changer une description de domaine, étant donnée une formule que nous voulons contracter.

Nous définissons donc une sémantique pour la contraction de théories et présentons également sa contrepartie syntaxique à travers des opérateurs de contraction. L'adéquation et la complétude de tels opérateurs par rapport à la sémantique ont été établies. En particulier, nous montrons que notre notion de modularité est une condition suffisante pour qu'une contraction soit réussie.

Dans ce travail nous avons utilisé une version faible de PDL, mais nos notions et résultats peuvent aussi bien s'appliquer à d'autres cadres logiques.

Index

If you don't find it in the index, look very carefully through the entire catalogue.
— Unknown, Sears, Roebuck, and Co. Consumer's Guide, 1897

“ Il en a rêvé, il l’a fait. ”